Künstliche Intelligenz: Einführung in Micro-Prolog

mc

3

Die Mikrocomputer-Zeitschrift

7 DM · 60 öS · 7 sfr. · März 1987

PC-Technik: Netzwerke

PC-Steckkarte: 12-Bit-D/A-Wandler selbstgebaut

Software: EGA-Routinen in Pascal

CP/M-68k auf dem mc-CP/M-Computer

Atari ST: Desk Accessories für Praktiker

Im Test: AT von Multitech

Symbolischer Debugger für PCs

Liebe Leser!

Netzwerke, das ist ein Thema, das zunehmend an Bedeutung gewinnt. Alle Kommunikationsprobleme innerhalb eines Betriebes sollen damit elegant und einfach zu lösen sein, das sagen die Hersteller. Mancher Käufer übersieht dann vielleicht doch, daß solch ein Netzwerk in die Kommunikationsumgebung innerhalb des Betriebes eingebettet werden muß. Dies sei hier nur deshalb festgestellt, weil mc vorwiegend technisch orientiert ist und über die Probleme, die mit der Kommunikationsstruktur in einem Betrieb zu tun haben (und damit auch bei der Implementation eines lokalen Netzwerkes berührt werden), nichts sagen wird. In dieser Ausgabe lesen Sie viel Technisches über Netzwerke. Wir haben uns dabei vorgenommen, das Thema so zu gestalten, daß man das Vokabularium handfest in den Griff bekommen kann. Vom Token-Ring bis zum ISO-Schichtenmodell.

Das Thema „Desktop Publishing" liegt in der Luft. Zunehmend erfährt man hier in der Bundesrepublik, daß in den USA schon kräftig in die Praxis eingestiegen wird. Eine große neue Welt der Gestaltung von Schriftgut und Grafik soll sich dort schon aufgetan haben. Faszinierend für alle, die Papieroberflächen mit ansprechenden und interessanten Botschaften versehen müssen. Allerdings, ein Heimcomputer und ein Matrixdrucker tun es dann nicht mehr. Dazu gehören mindestens ein 16-Bit-Computer, ein Drucker, dessen Output professionellen Ansprüchen genügen muß, und dazu noch Software, die das alles in Schwung bringt. Die Preise für so eine Station kann in der Regel zur Zeit nur eine professionell produzierende Firma bezahlen. Was mc betrifft, so wollen wir in dieser Ausgabe nur ganz bescheiden etwas über die Bearbeitung von Grafik und Texten mit der EGA-Karte bringen. Wenig, gemessen am Inhalt des Begriffes „Desktop Publishing" (der die Produktion ganzer Periodika umfaßt). Viel Nutzen aber für Sie, wenn Sie einen PC und eine entsprechende Karte besitzen oder sich überlegen, wie Sie heute mit erträglichen Kosten in die professionelle Gestaltung von Grafiken und Texten auf PCs einsteigen können. Denn was wir hier aufgrund der Fachkenntnisse unseres Autors bringen können, das sind grundlegende Beispiele in Pascal, die bis hin zum Einhängen mathematischer Funktionen in die Grafik reichen. Mit einem Unterprogramm, das das Ergebnis aufs Papier bringt. Und es soll noch nicht das letzte Wort gesprochen sein, ein Grafik-Editor soll folgen.

In mc sollen faszinierende Themen nicht verschwiegen werden, wenn möglich aber handfest zum Selbermachen und zum Selbstverwirklichen dienen können. Das haben wir zum Beispiel mit dem damals berühmten mc-CP/M-Computer bewiesen, der auch heute noch viele Kenner zum Nachbau reizt, weil er so ein ehrliches „Recheneisen" ist, das in vielen Fällen einfach als Computer ausreicht. Wer aber jetzt CP/M 2.2 verlassen muß, weil die Anforderungen gestiegen sind, der kann als weitere Ausbaustufe einen kompletten 68000-Rechner in das ECB-Gerät integrieren. Wir bei mc nehmen an, daß dies nicht nur die Besitzer des mc-CP/M-Computers preiswert in die nächsthöhere Computerklasse führt, sondern generell alle ECB-Rechnerbesitzer begeistert. Denn das Konzept erlaubt richtigen Parallel-Betrieb und man kann die Master-Rolle jedem der beiden Prozessoren, dem Z80 oder dem 68000 zuteilen. In dieser Ausgabe von mc wird die Implementation von CP/M-68k geschildert. Damit ist der Anschluß an die gewohnte professionelle Software-Umgebung gewonnen.

Es gibt ein Phänomen in der Computerszene, das uns bei mc – im Gegensatz zu unseren sonstigen Gewohnheiten – einen Blick in das statistische Jahrbuch der Insel Taiwan werfen ließ, uns veranlaßte, einige Auskünfte vom Taiwan Trade Service in Düsseldorf einzuholen und uns noch ein wenig herumhorchen ließ: Die explosionsartige Steigerung der Computer-Produkt-Importe aus diesem Land in die Bundesrepublik, die sich in einer ständigen Steigerung des Angebotes an preiswerten PCs niederschlägt. Dabei erfuhren wir als einen Grund, weshalb die Handbücher zu Produkten aus diesem Land so schmal sind: Man fürchtet Kopierer! Pikant vor dem Hintergrund, daß Taiwan gerade Kompatible so erfolgreich exportiert ...

Erfolgreiche Arbeit mit mc und Lesevergnügen wünscht

Ihr Ulrich Rockel

mc-inhalt

mc-editorial

Liebe Leser 3

mc-briefe 6

mc-info 10

Der Mensch im CIM-Mittelpunkt 64

Spruch des Monats 70

mc-bücher 36

Neues vom Buchmarkt 87

mc-grundlagen

Lokale Computernetzwerke 46
Mit ausführlicher Beschreibung von Netzwerktopologien und Übertragungstechniken

Rechner an der Scart-Buchse 51
Scart-Buchse ist nicht gleich Scart-Buchse. Wir zeigen, worauf beim Einsatz dieser Schnittstelle zu achten ist

Micro-Prolog 52
Für CP/M- und MS-DOS-Rechner gibt es mit Micro-Prolog eine leistungsfähige Implementation der Sprache Prolog

Interrupts bei Prozessoren 680XX 58
Die Prozessoren der 680XX-Familie kennen 230 Ausnahmen. Wir zeigen, wie sie richtig eingesetzt werden

Tips zu MASM und LINK 114

mc-soft

Multitasking und Desk Accessories 66
Desk Accessories stehen beim Atari ST immer auf Abruf bereit. mc zeigt, wie die nützlichen Programme geschrieben werden

Videodat-Empfänger für den C-64 72
Daten, die gleichzeitig mit dem Fernsehbild übertragen werden, erfaßt und verarbeitet ein C-64

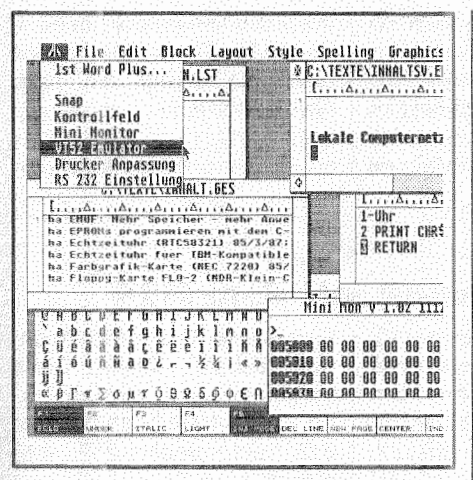
Grafik mit EGA 80
Durch die richtige Behandlung der EGA-Register wird die Arbeit mit der PC-Grafik zum Vergnügen

Lokale Computernetzwerke

Das Angebot an lokalen Computernetzwerken wächst ständig. Zeit also für jeden PC-Anwender, sich mit dieser Thematik auseinanderzusetzen, um Klarheit über die technischen Zusammenhänge zu erhalten. mc liefert die dazu notwendigen Informationen. **Seite 46**

Videodat-Empfänger C-64

Das Videodat-Verfahren wurde vom WDR-Computerclub zum Übertragen von Dateien über Fernsehsender erfunden. Am Beispiel, hier mit dem weitverbreiteten C-64 von Commodore, zeigt mc, wie die ASCII-Daten erfaßt und verarbeitet werden können. **Seite 72**

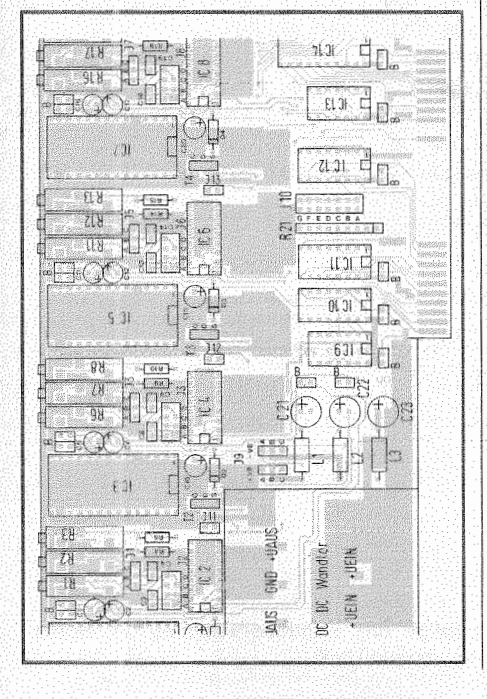


Desk Accessories für den Atari ST

Das Betriebssystem des Atari ST erlaubt den Aufruf von Programmen, während gerade ein anderes Programm läuft. mc gibt Tips zur Programmierung von Desk Accessories, mit denen man den Atari ST besser nutzen kann. **Seite 66**

D/A-Umsetzer für PCs

Sollen mit einem PC Geräte oder Prozesse über analoge Signale gesteuert werden, ist eine entsprechende Schnittstelle erforderlich. Wir zeigen, wie ein D/A-Umsetzer gebaut wird. **Seite 121**



Cursor ändern unter MS-DOS 86
Am Beispiel einer nützlichen Routine wird erklärt, wie Parameterübergaben an den Interrupt 10H funktionieren

Druckertestroutine mit Hex-Code-Anzeige 90
Bei der Behebung von Anpassungsproblemen mit Druckern hilft ein kleines Programm

Voreinstellung bei dBaseII ändern 91
So werden die Funktionstasten nach Wunsch belegt

Der mc-68-ECB und CP/M-68k 92
Das leistungsfähige Betriebssystem CP/M-68k wird auf dem mc-68-ECB-Rechner implementiert

Sichere Übertragung 104
Ein Programm in Turbo-Pascal erkennt Fehler in per Funk übertragenen Daten

Turbo-Pascal lädt DOS-3.3-Bilder 108
Ein Tip für alle APDOS-Anwender

mc-test

Maus mit Schlitzaugen 70
Ein „Rollkugeleingabegerät" für MS-DOS-Computer

Turbo-Symbolic-Debugger beschleunigt Fehlersuche 108
Darauf haben viele Turbo-Pascal-Programmierer gewartet

AT-Kompatibler aus Taiwan 110
Dieser Rechner kann sparsamen Anwendern empfohlen werden

Periscope 112
Ein symbolischer Debugger für Assemblerprogrammierer

MS-DOS als Beifahrer 116
Der Amiga von Commodore tanzt auf zwei Hochzeiten gleichzeitig

Transputer konkret 118
30 MIPS mit Transputern

mc-hard

D/A-Umsetzer für den PC 121
Unser Selbstbauprojekt in dieser Ausgabe ist für alle interessant, die mit einem MS-DOS-Rechner analoge Größen steuern wollen

mc-markt 130

mc-vorschau 174

Impressum 172

Dauerton

Für das in mc 12/86 abgedruckte Listing eines Sequencer-Programmes möchte ich einen Verbesserungsvorschlag machen. Die Transmit-Routine läßt sich zwar durch Drücken einer beliebigen Taste abbrechen, die Töne, die bis zu diesem Zeitpunkt aber noch nicht ausgeschaltet wurden, klingen auf dem Synthesizer weiter und sind dann meist nur noch durch das berühmte Knöpfchen auf der Rückseite zum Verstummen zu bringen. Folgenden kleine Routine schafft Abhilfe:

```
Allout LDA #$B0
       JSR ACIAOUT
       LDA #$7B
       JSR ACIAOUT
       LDA #$00
       JSR ACIAOUT
```

Diese Routine sendet den Befehl B0 7B 00 an den Synthesizer, der daraufhin alle Töne ausschaltet. Sie sollte im Assembler-Listing hinter Zeile 337 eingefügt werden.

Jost Langenfeld
2370 Rendsburg

Betrug

Vor kurzem gab ich eine Kleinanzeige in einer Elektronikzeitschrift (bei mc) mit folgender Aufmachung auf: „Selbstgeschriebenes Programm für 20 DM abzugeben. Probeausdruck anfordern."

Nach dem Erscheinen wunderte ich mich darüber, daß keine Anfragen kamen. Einige Zeit später sprach mich dann ein Postbeamter (der mich persönlich kannte) darauf an, daß meine Post nach Düsseldorf nachgesandt wird. Wie sich später herausstellte, wurde von einer fremden Person ein Post-Nachsendeantrag an mein Postamt geschickt, der mit meinem Namen und Adresse versehen war. Die Adresse war ja in der Anzeige zu finden. Vermutlich hat es der Betrüger oder die Betrügerin auf die 20 DM abgesehen. Die Post kann gar nichts dagegen tun, weil sie der amtlichen Schweigepflicht untersteht - sie hätte es mir nicht einmal sagen dürfen!
Daher ein paar Tips an alle, die eine Klein-Anzeige privat aufgeben wollen bzw. die darauf antworten:

- Bei Geld-Beträgen nie eine Adresse sondern nur die Telefon-Nummer angeben.
- Niemals Bargeld verschikken, besser einen Scheck oder direkt auf ein Konto überweisen.
- Wenn Schecks verschickt werden, immer die komplette Adresse des Empfängers und „Nur zur Verrechnung" auf den Scheckkopf schreiben.

Adresse ist der Redaktion bekannt

Grafik und Olivetti M24

Das Problem der inversen Textdarstellung in der Grafik des Olivetti M24 unter Turbo-Pascal konnte unter Mithilfe von Herrn Oker (dessen Leserzuschrift Sie an mich weiterleiteten) geklärt werden (Leserbrief in mc 12/86).
Text wird auch im Grafikmodus mit Hilfe der Unterbrechung 10h geschrieben (mit AH = 9). Dabei steht im Register BH das Attribut des Zeichens (vgl. auch mc 11/86, S. 72 ff). Im IBM-Grafikmodus wird dieses Attribut-Byte allem Anschein nach ignoriert, während es bei der hochauflösenden Grafik des M24, zumindest für die inverse Textdarstellung, noch abgefragt wird. Durch Angabe von TextColor(White) und TextBackground(Black) vor der Umschaltung durch HiRes, kann das Attribut-Byte so eingestellt werden, daß die Schrift nicht mehr invers erscheint. Wenn TextColor und TextBackground nicht vereinbart werden, ist das Verhalten von der Versionsnummer des ROM-BIOS abhängig. Während bei Version 1.1 die Schrift spätestens beim zweiten Aufruf normal erscheint, wird sie bei der Version 1.36 immer invers dargestellt. Dieses Problem tritt meines Erachtens im übrigen auch bei der in mc 11/86, S. 110 ff. vorgeschlagenen Implementierung der hochauflösenden Grafik auf, wenn nicht gleichzeitig die neue Gwrite-Prozedur verwendet wird.

Jörg Klauer
5431 Mogendorf

Hohe Auflösung mit dem M24

Die in mc 11/86 (Seite 110, Bild 1) abgebildete externe Maschinenroutine führt beim Aufruf aus einer Turbo-Pascal-Prozedur zum Systemabsturz. Ursache ist die fehlende Stapel-Sicherung des Basis-Seitenregisters BP.
Eine kleine Programm-Änderung schafft Abhilfe. Diese Vorgehensweise wird im Turbo-Pascal-Handbuch (Version 3.0) auf Seite 252 beschrieben.

Manfred Weis
8900 Augsburg

Änderung der Maschinenroutine

Alt:		Neu:	
plot:	mov bp,sp	plot:	push bp
	mov si,[bp+6]		mov bp,sp
	...		mov si,[bp+8]
	...		...
	mov di,[bp+4]		mov di,[bp+6]
	...		...
	mov cx,[bp+2]		mov cx,[bp+4]
	...		...
endpl:	ret 6	endpl:	pop bp
			ret 6

Linienalgorithmus

Wie im Leserbrief von Herr Kling (mc 11/86) richtig festgestellt wurde, ist der Linienalgorithmus in mc 9/86, Seite 12 umständlich. Dies trifft auch auf den Linienalgorithmus im Artikel „Grafik Grundlagen" in mc 2/86, Seite 79 zu. In beiden Fällen wird der Algorithmus durch die Randüberwachung aufgebläht.
Steht ein LINE oder DRAW-Befehl zur Verfügung, so kann mit der Funktion CLIP (*Bild*) die Randüberwachung sehr effektiv durchgeführt werden.

Gerhard Röhner
6100 Darmstadt

```
FUNCTION CLIP(X1,Y1:INTEGER; VAR X2,Y2:INTEGER):BOOLEAN;
BEGIN
  IF (Y2<WYMIN) AND (WYMIN<=Y1) THEN
    BEGIN X2:=X1+ROUND((WYMIN-Y1)/(Y2-Y1)*(X2-X1)); Y2:=WYMIN END ELSE
  IF (Y1<=WYMAX) AND (WYMAX<Y2) THEN
    BEGIN X2:=X1+ROUND((WYMAX-Y1)/(Y2-Y1)*(X2-X1)); Y2:=WYMAX END;
  IF (X2<WXMIN) AND (WXMIN<=X1) THEN
    BEGIN Y2:=Y1+ROUND((WXMIN-X1)/(X2-X1)*(Y2-Y1)); X2:=WXMIN END ELSE
  IF (X1<=WXMAX) AND (WXMAX<X2) THEN
    BEGIN Y2:=Y1+ROUND((WXMAX-X1)/(X2-X1)*(Y2-Y1)); X2:=WXMAX END;
  CLIP:=(WXMIN<=X2) AND (X2<=WXMAX) AND (WYMIN<=Y2) AND (Y2<=WYMAX)
END;

PROCEDURE LINIE(X1,Y1,X2,Y2:INTEGER);
BEGIN
  IF CLIP(X1,Y1,X2,Y2) AND CLIP(X2,Y2,X1,Y1)
    THEN LINE(X1,Y1,X2,Y2);
END;
```

Diese Clip-Routine erledigt die Randüberwachung in Grafik-Progammen schnell und unkompliziert

Und es stimmt doch!

Das in mc 11/86 abgedruckte Programm erstellt den richtigen EAN-Code. Es ist Dipl.-Ing. G. Skrzeba offensichtlich entgangen, daß es sich bei der sogenannten Prüfziffer um eine Eingabegröße handelt. Sie dient zum Verschlüsseln des Landes, in dem sich die Vertriebsorganisation eines Produktes befindet, so bedeutet eine 4, daß der Vertrieb dieses Artikels in Deutschland liegt. Hätte Dipl.-Ing. G. Skrzeba sich mal die Mühe gemacht, die europäische Artikelnummer von einem in der Nähe befindlichen Artikel mitsamt der Prüfziffer (wie es das Programm vorsieht) einzugeben, hätte er/sie vielleicht erstaunt festgestellt, daß dies nicht ein optisch ähnlicher Code, sondern eine genaue Reproduktion dessen ist, was sich auf dem Artikel befindet (*Bild*). Sonst dürfte etwa auf der Shampoo-Flasche eines großen deutschen Hautcreme- und Pflegemittelherstellers nicht die Ziffernkombination 005800148194 mit der Prüfziffer 4 stehen, sondern als Prüfziffer müsste 6 auftauchen.

Wolfgang Klinke
7505 Ettlingen 4

Etikett eines Brotes aus Frankreich (Prüfziffer 3; nach Skrzeba: 5)

Mein Leserbrief „EAN-Codes drucken" in mc 1/87, Erwiderung von Herrn Klinke vom 3. 1. 1987

Und es stimmt doch nicht!

... was Herr Wolfgang Klinke aus Ettlingen in seiner Zuschrift auf meinen o.a. Leserbrief über die Europäische Artikelnummer (EAN) und insbesondere deren Prüfziffer ausführt.

In der Bundesrepublik Deutschland ist die Zentrale für Coorganisation, Gesellschaft zur Rationalisierung des Informationsaustausches zwischen Handel und Industrie mbH (CCG) für die Organisation der EAN zuständig. Aus deren Schrift „Coorganisation EAN Teil 1, Die Europäische Artikelnumerierung in der Bundesrepublik Deutschland" ist zu entnehmen, daß sich die 13stellige EAN (Normalnummer) von links nach rechts so aufbaut:

Land 12, Betriebsnr. 12345, individuelle Artikelnr. 12345, Prüfziffer 1.

Die ersten beiden Stellen (von links) stellen also die Länderkennung dar, die 13. Stelle (ganz rechts) die Prüfziffer. Sowohl der Autor des Artikels „EAN-Codes drucken" in mc 11/86 als auch Herr Klinke wollen jedoch die erste Stelle von links als Prüfziffer gewertet wissen. Im übrigen lauten die wichtigsten Länderkennzeichen in Europa:

30-37	Frankreich
40-43	Bundesrepublik Deutschland
50	Großbritannien u. Irland
54	Belgien u. Luxemburg
57	Dänemark
76	Schweiz
80-81	Italien
84	Spanien

Aus der genannten Schrift ergibt sich auch, daß es sich bei der Prüfziffer *nicht* um eine Eingabegröße handelt, wie Herr Klinke meint, sondern daß diese tatsächlich nach dem Verfahren Modulo 10 errechnet wird, wie ich in meiner Leserzuschrift in mc 1/87 an einem Beispiel demonstriert habe.

Nimmt man das Beispiel aus dem Brief von Herrn Klinke in der *richtigen Reihenfolge* (die ersten 12 Stellen von links), so ergibt sich nach dem Verfahren Modulo 10 die Prüfziffer 4. Diese ist auch auf dem Originaletikett und dem Musterabdruck rechts als 13. Stelle angegeben.

Wer es ganz genau wissen möchte, kann sich die genannte Broschüre sowie deren Teil 2 „Der EAN-Strichcode" zum Preis von jeweils 24,– DM bei folgender Adresse bestellen: CCG, Postfach 190209, 5000 Köln 1.

Dipl.-Ing. G. Skrzeba,
7900 Ulm

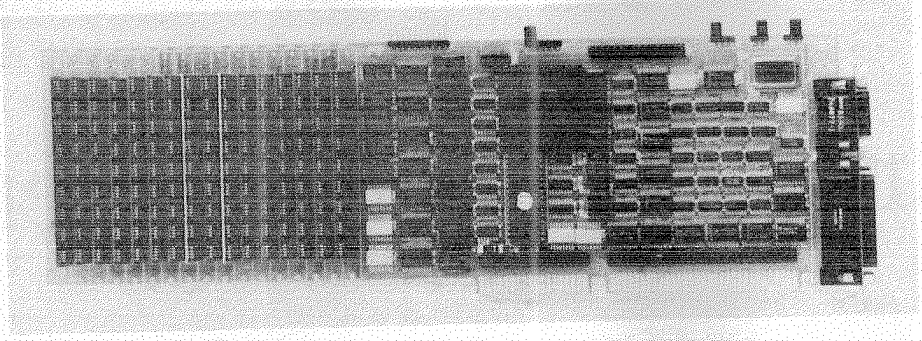
Leistungs-explosion

Passend zu Ihren in letzter Zeit immer häufiger erscheinenden Artikeln über die Leistungsexplosion bei Mikroprozessoren habe ich gerade eben meinem Modem via Satellit direkt aus Silicon Valley eine brandheiße Nachricht entlocken können, die ich hiermit unverzüglich und exclusiv an Sie weitergeben möchte:

Dem Mikroprozessor-Hersteller LETNI ist es nun endlich gelungen, das erste Exemplar eines optischen Mikroprozessors im Labor herzustellen. Nach außen hin verhält er sich genau so, wie ein ganz normaler Halbleiterbaustein mit TTL-kompatiblen Ein- und Ausgängen, aber intern werden alle Informationen mit Hilfe von elektromagnetischen Wellen codiert und mit Lichtgeschwindigkeit weiterverarbeitet. Aus Geheimhaltungsgründen war über den inneren Aufbau des Chips leider nicht mehr zu erfahren, aber eine kurze Beschreibung seiner Fähigkeiten wurde mir freundlicherweise gegeben.

Demnach stellt der M1097373, so heißt der Mikroprozessor, einen echten 256-Bit-Prozessor dar, bei dem sowohl Adressen als auch Daten 256 Bit breit sind. Er hat 1024 Allzweck-Register und einen orthogonalen Befehlssatz mit 47 verschiedenen Adressierarten. Er unterstützt alle bekannten höheren Programmiersprachen, aber optimal ausgelegt ist er erst für die Programmiersprachen der sechsten Generation. Vier davon sind übrigens bereits auf dem Chip im Pico-ROM sowohl in der Interpreter-, als auch in der Compiler-Version integriert. Als Leckerbissen für CAD-Anwender wurden 16 parallel arbeitende Arithmetik-Prozessoren, die zum Beispiel zur Berechnung von Transformationen benützt werden können, sowie ein Grafik-Prozessor mit 64 MByte Bildwiederholspeicher mitintegriert. Die acht Signalprozessoren, die für Meßtechniker interessant sein dürften, erwähne ich hier nur der Vollständigkeit halber. Eine weitere Spezialität stellt auch der eingebaute LASER dar, der für das Schreiben und Lesen einer über dem Prozessor rotierenden optischen Disk gedacht ist. Die Vielzahl der vorhandenen Schnittstellen erscheint angesichts dieser Sachen trivial. Fast vergessen hätte ich auch noch das mitintegrierte RAM mit einem Umfang von 4 GByte und das schnelle Cache, das 16 MByte umfaßt. Das wichtigste Feature jedoch, das dem M1097373 einen revolutio-nären Markterfolg bescheren wird, ist seine Kompatibilität zum ABACUS.

Hermann Maser
7300 Esslingen



Kombiniert Grafik, Speicher und Schnittstellen auf einer Karte: All-Aboard-286

EGA und 16 MByte RAM kombiniert

Die Multifunktions-/Grafik-Karte All-Aboard-286 von IDEAssociates, München, wurde speziell für den IBM XT-286 entwickelt. Sie ist jedoch auch für den AT geeignet. Durch die fast ausschließliche Bestückung mit Bausteinen in Oberflächenmontagetechnik ist es gelungen, alle wichtigen Funktionen auf einer Karte zu integrieren, statt wie bisher auf drei oder mehr Karten. All-Board-286 kann mit bis zu 16 MByte Speicher bestückt werden. Laut IDEAssociates unterstützt die Karte bereits die Möglichkeiten, die DOS 5.0 haben wird.

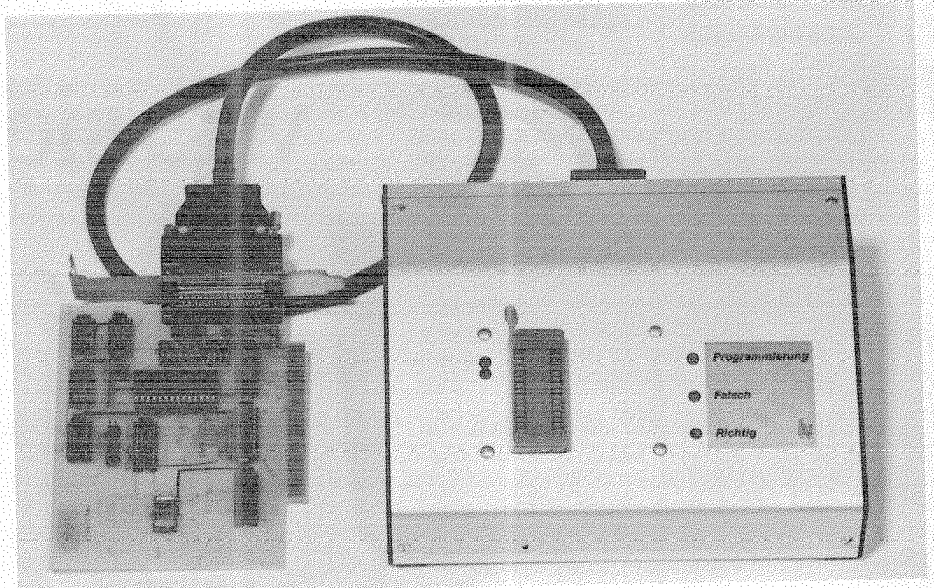
Der Grafikteil kann im CGA-, MDA- und EGA-Modus betrieben werden. Zusätzlich hat die Karte eine serielle und eine parallele Schnittstelle.

EPROM-Programmiersystem

Ein EPROM-Programmiersystem mit einem eigenen Prozessor, der als Sub-System an MS-DOS-Rechner und CP/M-Systeme angeschlossen werden kann, hat List & Niemann, Wiesbaden, entwikkelt. Fast alle EPROMs der Serie 25xxx und 27xxx können vom EPROG-3S programmiert werden. Schnellprogrammieralgorithmen der Hersteller Intel, Fujitsu, AMD und Texas Instruments werden vom EPROG-3S unterstützt. Das Programmiersystem ist durch die Bedienerführung mit Menüs einfach handhabbar. Es kann Dateien lesen, schreiben und umbenennen. Der Anwender kann Speicherbereiche editieren, durchsuchen, füllen und ausdrucken.

Ist im PC kein Steckplatz mehr frei, so läßt sich der EPROG-3S über COM1 betreiben.

Für die Einchip-Mikroprozessoren 8744, 8751 und 87C51 gibt es einen zusätzlichen Programmier-Adapter.



Die PC-Version des EPROG-3S

mc-68000 an der Volkshochschule

In einem Lehr- und Übungskurs der Münchner Volkshochschule wird der Einsatz der Prozessoren 68000 und 68010 am Beispiel des mc-68000-Computers beschrieben. Der Kurs umfaßt die Funktionsbeschreibung und den Aufbau des Rechners sowie eine Einführung in die Betriebssystem-Software. Der Kurs kostet 290 DM und dauert 60 Stunden.

Commodore im Aufwind

Der Umsatz der Commodore Büromaschinen GmbH (Frankfurt) stieg im abgelaufenen Geschäftsjahr um 3,5 Prozent von 926,2 Millionen DM auf 958,9 Millionen DM. Der Gewinnüberschuß beläuft sich auf 9,9 Millionen DM. Innerhalb des weltweiten Konzerns macht der deutsche Umsatz 23 Prozent aus.

Mehr zum gleichen Preis

Nach der Device „Darf es etwas mehr sein" liefert Victor Technologies, Frankfurt, das AT-kompatible System V286 jetzt mit 30 MByte und 60 MByte Festplatte zum unveränderten Preis aus. Die Plattenkapazität wurde durch ein neues Aufzeichnungsverfahren um 50% gesteigert. Für das neue Verfahren sind nur Platten mit rauschfreier Oberfläche und rauscharmen Lesekanälen geeignet.

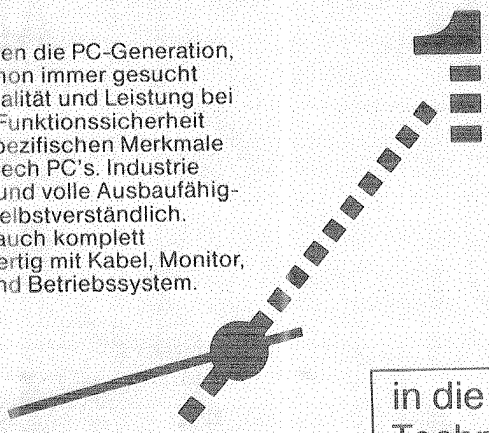
GEM-Software mit Turbo erstellt

Für Leute, die GEM-Software in Turbo-Pascal erstellen wollen, hat die Karlsruher Firma Bechtold Ingenieurgesellschaft ein Softwarepaket entwickelt. Es läuft auf IBM-PC-kompatiblen Computern und heißt Turbo-GEM-Toolkit. Das Paket enthält den Quelltext von mehr als 180 Prozeduren und Funktionen, mit deren Hilfe man GEM-Schnittstellen programmieren kann. Sie entsprechen den C-GEM-Bindings von Digital Research. Daneben gehört zu diesem Produkt ein Editor zur Erstellung von GEM-Resource-Dateien und – wichtig für den Einstieg – ein Programmbeispiel zur Anwendung des Toolkits. Preis: 285 DM.

Multitech

...und weiter geht es

Wir haben die PC-Generation, die Sie schon immer gesucht haben: Qualität und Leistung bei optimaler Funktionssicherheit sind die spezifischen Merkmale aller Multitech PC's. Industrie Standard und volle Ausbaufähigkeit sind selbstverständlich. Natürlich auch komplett anschlußfertig mit Kabel, Monitor, Tastatur und Betriebssystem.



ACCEL 900 AT und die bewährten Produkte in ausgereifter Technik. Jetzt bei Multitech neu im Programm:
- Laser-Printer
- Comocolan Netzwerk

Die neue Technologie des MPC 1100 mit 32 BIT-Rechner zu erschwinglichen Preisen.

in die Technologie des **80 386**

Wir lüften das Geheimnis!

MPC 1100

Hannover Messe'87

CeBIT – 4.–11. MÄRZ

wir stellen aus:
Halle 6, Stand 56/60

C 46153

Generalimporteur: CE-TEC International

Bitte Händlernachweis anfordern.

CE-TEC Trading GmbH · Kornkamp 4 · D-2070 Ahrensburg
Telefon: 04102/49 01-0 · Telex: 2 189 875 · Telefax: 04102/49 01 38

mc-info

UART vereinfacht serielle PC-Schnittstellen

Speziell für den Einsatz in IBM-PC-kompatiblen Computern hat Intel einen neuen UART (Universal Asynchronous Receiver Transmitter) entwickelt. Dadurch, daß man sich auf die Ansteuerleitungen beschränkt hat, die für den PC-Bus erforderlich sind, sank die Zahl der Anschlüsse gegenüber dem Standard-Baustein 8250 von 40 auf 28. Das senkt nicht nur die Herstellungskosten für den Chip, sondern auch den Aufwand für den Hardware-Entwickler: Zusätzliche TTL-Logik ist nämlich kaum mehr erforderlich. Insbesondere für Systeme, die mit mehr als 10 MHz betrieben werden, vermindert sich der Aufwand beträchtlich. Zusammen mit dem Baustein 89024 kann man mit dem neuen UART sehr preiswert komplette Modems für PCs realisieren.

Open-Access-II umtauschen

Wer bislang mit dem integrierten Paket Open-Access gearbeitet hat, kann jetzt im Rahmen einer Umtausch-Aktion auf Open-Access-II umsteigen. Die neue Version kostet für Umsteiger 495 DM.

C-Compiler für 68000-Familie

Einen C-Compiler für die Prozessoren der Familie 68000, der dem einschlägigen ANSI-Standard entspricht, hat die schweizerische Retis Realtime Software AG, Aarau, entwickelt. Er läuft unter dem Betriebssystem Versados, ist aber auch als Cross-Compiler für IBM-PCs (PC-DOS), VAX- (VMS) und 68000-Unix-Rechner erhältlich. Der Compiler erzeugt Code für die Prozessoren 68000, 68010 und 68020 und unterstützt den erweiterten Befehlssatz des Arithmetik-Chips 68881.

Sein Objektcode läßt sich in Befehls- und Datensegmente zur Ablage in ROM und RAM trennen. Er kann im Motorola-S-Record-Format oder anderen Formaten an Emulatoren, Programmiergeräte oder den Zielrechner geschickt werden. Der generierte Assemblercode läßt sich zusammen mit dem Quelltext als gemeinsames Listing ausdrucken.

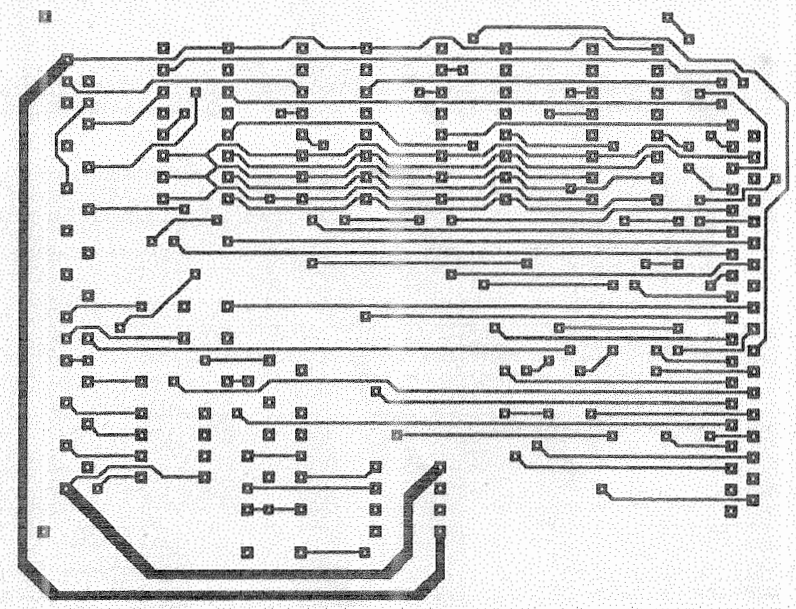
Kompatibler von Atari

Atari wird zur CeBit in Hannover einen IBM-PC-kompatiblen Computer vorstellen, der mit EGA-Karte und EGA-Monitor unter 2000 DM kosten soll. Der Atari-PC ist mit der CPU 8088 ausgestattet, die wahlweise mit 4,77 oder 8 MHz betrieben werden kann. Ein Sockel für den Coprozessor 8087 ist vorhanden, Steckplätze für Erweiterungskarten bietet der Computer im ST-Design nicht. Zum Ausbau des Computers soll es eine Expansionbox geben, die auf das Grundgehäuse gestellt werden soll. Als Betriebssystem soll der Atari-PC DOS 3.2 und GEM erhalten. In welchem Umfang GEM-Software mitgeliefert werden soll, war bei Redaktionsschluß noch nicht bekannt. Eine Maus zur Bedienung der GEM-Programme wird mitgeliefert. Die Grundversion enthält 512 KByte RAM, 256 KByte RAM für die EGA-Karte, eine serielle und eine parallele Schnittstelle sowie einen Mausanschluß. Der Atari-PC ist in der Grundversion mit der EGA-Karte ausgerüstet.

Vom Schaltplan zum Layout

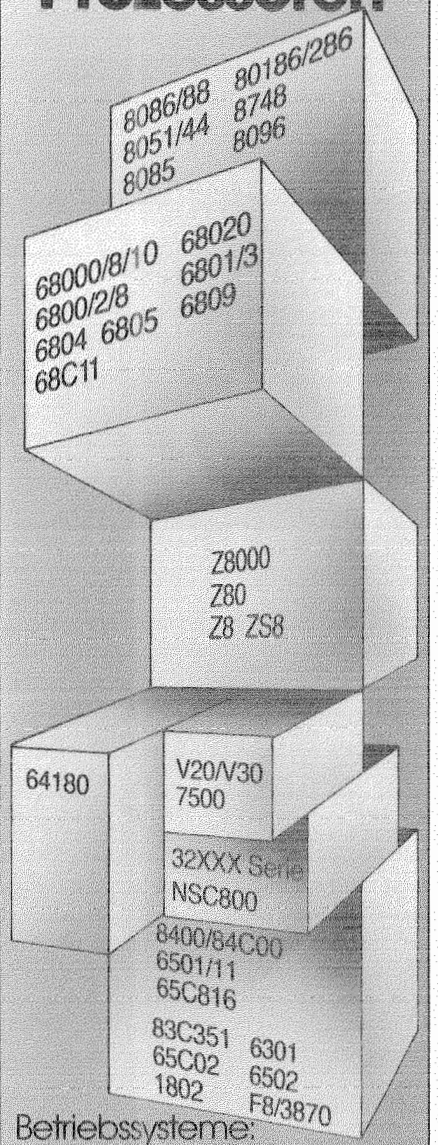
Die Arbeit des Schaltungsentwicklers erleichtern die beiden Programme Franzis-CAD und Auto-Router 3, die Franzis-Software für IBM-PC-kompatible Computer anbietet. Franzis-CAD (Preis: 495 DM) ist zwar ein universelles Zeichenprogramm, wegen seiner umfangreichen Bibliothek mit vordefinierten Symbolen für elektronische Bauelemente profitieren aber insbesondere Elektroniker von ihm. Das Platinenentflechtungsprogramm Auto-Router gibt es schon etwas länger, die neue Version 3 (Preis: 764 DM) kann aber mit vielen neuen Funktionen und einer noch bequemer zu handhabenden Benutzeroberfläche aufwarten.

Wer mit dem CAD-Programm eine Schaltzeichnung eingibt, muß für den Auto-Router die Verbindungsliste nicht gesondert erstellen: Ein Konvertierungsprogramm nimmt ihm diese Arbeit ab. Der Vorteil dabei besteht nicht nur darin, daß man sich einen Arbeitsgang spart, sondern betrifft vor allem auch die „durchgängige" Dokumentation vom Schaltplan bis zum Layout und zum Bestückungsplan (den der Auto-Router ebenfalls liefert). Beide Programme geben das Ergebnis ihrer Bemühungen auf fast jedem Plotter aus. Auto-Router 3 steuert zusätzlich beliebige Drucker an. Zur Eingabe benötigt das Zeichenprogramm eine Maus.



Die Schaltzeichnung muß man noch selbst eingeben, den Rest von der Dokumentation bis hin zur Ausgabe der Layout-Vorlage erledigen die beiden Programme Franzis-CAD und Auto-Router 3: Hier die Lötseite eines automatisch berechneten Layouts, ausgegeben mit einem 24-Nadel-Drucker

CROSS ASSEMBLER und COMPILER von 2500AD SOFTWARE INC. für die Prozessoren



Betriebssysteme:
MSDOS · UNIX / VMS · CP/M
COMPILER für C und PASCAL

Händlerkonditionen sowie ausführliche Unterlagen und Preise auf Anfrage!

allmos electronic

Allmos Electronic GmbH
Fraunhoferstraße 11A · 8033 Martinsried
Tlx. 5215111 alec d · Tel. 089/8572086-89

LASER-
DRUCKER
profis

die Leute von laserteam:
* Über 300 Laser installiert
* viele Anpassungen und Treiber
* eigene Software für Laser

für jede Anforderung
finden Sie
eine praktikable Lösung:
* Textverarbeitung
* Formularerstellung
* Plotten
* Desk-Top-Publishing

Wir zeigen Neuheiten:
in Halle 4
1.OG Stand A56
* die neuen Laserdrucker,
* den neuen Scanner
* die PC's + AT's
von Canon

HANNOVER MESSE
CeBIT '87
Welt-Centrum Büro · Information · Telekommunikation
4. – 11. MÄRZ 1987

Achtung
Fachhändler:
Mit dem Starterpack von laser team halten Sie den problemlosen Einstieg in Ihren Verkaufserfolg in Händen.
Informieren Sie sich oder besuchen sie uns.

laser team
6070 Langen Gartenstraße 100
Telefon 06103 22037-39

PC wird zum „Geschoß"

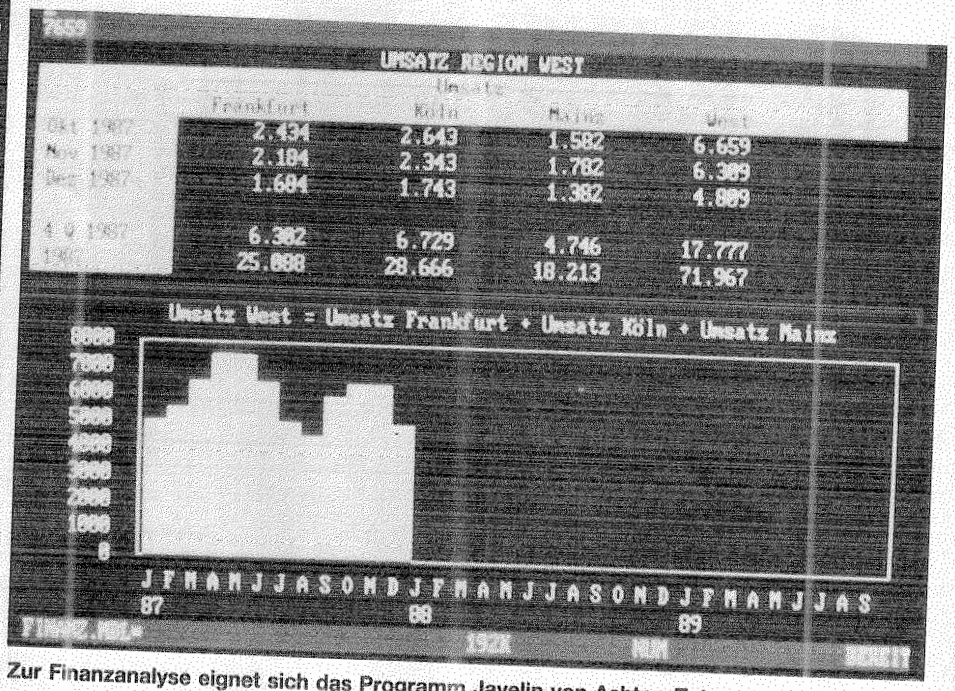
Wer seinen lahmen PC zum superschnellen Geschoß umrüsten will, kann dies jetzt mit einer Zusatzplatine der Firma Wave Mate, Unterschleißheim, tun. Bullet-286 nennt sich die Einheit, die mit dem 80286 und 1 MByte RAM bestückt ist. Ein normaler PC wird damit um den Faktor 9 schneller.
Mit den üblichen Beschleunigungskarten hat die Bullet-286 wenig gemein. Sie stellt nämlich eine eigenständige Prozessorplatine dar, die in etwa 20 Minuten gegen die Original-Platine ausgetauscht werden kann. Alle Anschlüsse und Abmessungen sind mit denen des IBM-PC/XT identisch. Die neue Konfiguration ist sofort betriebsbereit. Änderungen am Betriebssystem sind nicht erforderlich. Die Taktfrequenz wird vom Anwender bestimmt. Wählbar sind 8, 10 oder 12,5 MHz. Der Preis für das komplette „Tuning"-Paket Bullet-286 beträgt rund 2600 DM.

GfA-Basic für jeden

Das in mc 2/87 beschriebene GfA-Basic für den Atari ST wird nicht, wie wir irrtümlich berichteten, auf einer doppelseitigen Diskette, sondern auf einer einseitig formatierten Diskette verkauft. Somit kann jeder ST-Anwender dieses Basic, das bislang weltweit 16 000mal verkauft wurde, auch mit einem einseitigen Floppy-Laufwerk einsetzen. GfA-Basic kann damit zum Standard-Basic der ST-Anwender werden.

Software des Jahres in deutsch

Journalisten aus sieben Ländern haben das Programm Javelin von Ashton-Tate kürzlich zur Software des Jahres gewählt. Javelin ist ein Kalkulationsprogramm, das sich von seinen Konkurrenten grundlegend unterscheidet. Es ist ab sofort in deutsch erhältlich.
Insbesondere im Bereich der Finanzanalyse zeigt das Programm neue Wege auf. Informationen werden zentral als Variablen abgelegt. Zugriff darauf hat man über zehn sogenannte „Views" wie Arbeitsblatt, Flußdiagramm, Formeln, Struktur oder Grafik. Ganz gleich, welche „Sicht" man wählt, man bearbeitet immer denselben Datenbestand. Das Programm arbeitet mit zahlreichen Speichererweiterungen und Grafikkarten zusammen und kostet 2450 DM. Wer schon die englische Version hat, kann für etwa 560 DM die erforderliche Software zur Umstellung auf die deutsche Sprache erwerben.



Zur Finanzanalyse eignet sich das Programm Javelin von Ashton-Tate

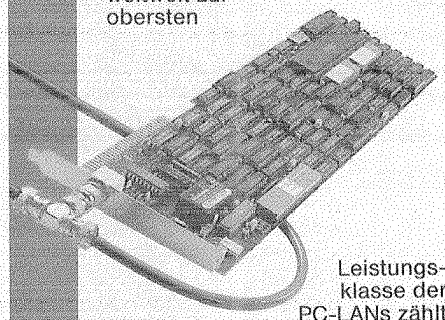
Für eilige Anfragen: mc-Kontaktkarten verwenden

Geschwindigkeit.
Intelligenz.
Leistung.

Ethernet LAN
SK-NET

SK-NET ist High-Tech made in Germany

Jeder Leistungsvergleich wird den kritischen PC-Anwender davon überzeugen, daß SK-NET weltweit zur obersten



Leistungsklasse der PC-LANs zählt – und sich innerhalb dieser Elite durch ein ausgezeichnetes Preis-Leistungsverhältnis abhebt...

SK-NET, die Kombination der Superlative:

- Ethernet-Standard: 10 Mio. Bits/sec
- Controller-Intelligenz mit MC 68000
- Preiswerte Cheapernet-Verkabelung
- Performance mit NetWare von NOVELL
- und IBM PC-Network
- Kompatibel zur MS-DOS-Welt.

Sprechen Sie mit uns. Wir reden Deutsch.
Und vernetzen weltweit.

Schneider & Koch & Co. GmbH
DATENSYSTEME
Haid-und-Neu-Str. 7–9
D-7500 Karlsruhe 1
Telefon 07 21 / 6 05 21

MS-DOS, IBM PC-Network, Ethernet und NetWare sind eingetr. Warenzeichen.

mc-info

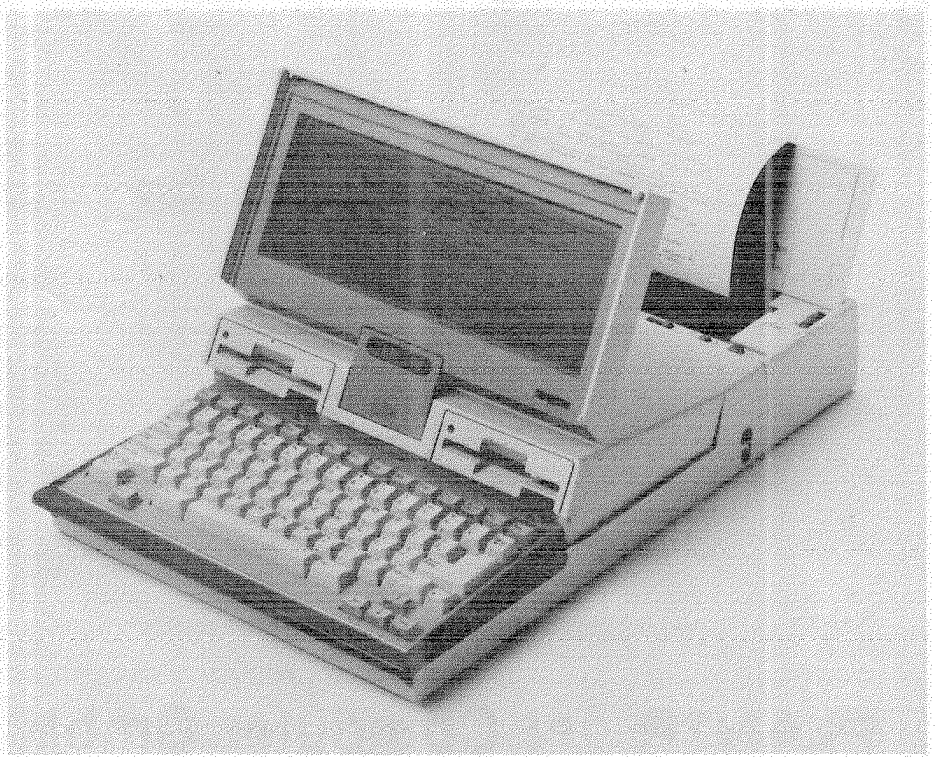
Mobiles 1200-Bd-Modem bekommt FTZ-Nummer

Wie ein Akustikkoppler ist das 1200-Bd-Vollduplex-Modem Adam 1200 von CTK, Bergisch-Gladbach, zu betreiben. Dennoch ist diese Bezeichnung nicht korrekt, denn das Gerät wird induktiv an das Telefon angekoppelt. Umgebungsgeräusche haben deshalb keinen Einfluß auf die Übertragung.
Dieses Modem, kann jetzt offiziell am Telefonnetz der Deutschen Bundespost betrieben werden: Seit Dezember hat es die dazu erforderliche FTZ-Nummer (schon vorher war die Zulassung in Österreich erteilt worden). Man hat jetzt die Wahl, statt der Postmodems D1200-S10 oder D1200-S12, die für teures Geld zu mieten sind, ein eigenes Modem einzusetzen. Das Gerät kostet gut 2100 DM.

IBM-PC-AP jetzt auch in Deutschland

Den tragbaren Personal Computer PC-AP von IBM, der in den USA unter dem Namen „Convertible" bekannt ist, wird es bald auch in Deutschland geben. Erste Auslieferungen für den „Advanced Portable" mit deutscher Tastatur sind für April geplant.
Der AP enthält als CPU die CMOS-Version des 8088, die ebenso wie im großen Bruder mit 4,77 MHz läuft. Als Bildschirm dient ein grafikfähiges Flüssigkristall-Display mit 25 Zeilen zu je 80 Zeichen. Der Anschluß eines externen Bildschirms ist nicht vorgesehen. Zwei 3,5-Zoll-Disketten-Laufwerke stellen den externen Massenspeicher dar. Versorgt wird das Gerät von Akkus oder einem separaten Netzteil. Bei Netzbetrieb werden die Akkus automatisch geladen. Das Gerät wiegt einschließlich Batterien 5,5 kg und mißt 360 mm * 309 mm * 67 mm (Länge * Breite * Höhe). Je nach Softwareausstattung liegt sein Preis zwischen etwa 5800 und 6100 DM.

Auf den AP läßt sich von hinten ein Drukker aufstecken, der nach dem Thermo-Transfer-Verfahren arbeitet. Er läßt sich mit Thermo- oder Normalpapier betreiben und kostet knapp 1000 DM. Als weiteres Zubehör gibt es einen Adapter mit paralleler und serieller Schnittstelle (650 DM).



Bald in deutscher Ausführung zu haben: der IBM-PC-AP

Heim-Amiga und Profi-Amiga

Commodore hat die Erfahrungen, die mit dem Amiga 1000 gesammelt wurden, in die Entwicklung von zwei neuen Produkten der Amiga-Familie einfließen lassen. Rechtzeitig zur CeBit in Hannover kann das Unternehmen den Amiga 500 und den Amiga 2000 vorstellen. Das kleinere System ist für den Heimbereich gedacht, das größere ist für den professionellen Einsatz (auch als PC-kompatibler) konzipiert.

Der Amiga 500 ist das Resultat, das bei dem Versuch herauskam, einige Wünsche der am Amiga Interessierten zu verwirklichen. Für 1298 DM (preiswerter), mit integrierter Tastatur (kleiner), 512 KByte RAM (voll einsatzfähig) und einem eingebauten Floppy-Laufwerk (kompatibel) gibt es jetzt einen Amiga. Der Amiga 500 entspricht weitgehend dem Amiga 1000, ist voll kompatibel zu diesem System, zielt deutlich auf den Heimcomputermarkt und konnte durch weiter gesteigerte Integration der Bauelemente in einem Tastaturgehäuse untergebracht werden. Zahlreiche TTL-Funktionen wurden in dem vierten Custom-Chip „Garry" zusammengefaßt. Auch der für DMA, Grafik- und Bildmanipulationen zuständige Chip „Agnus" wurde erweitert und heißt nun „Fat Agnus". Commodore gibt an, daß dieser Computer das Ergebnis zweijähriger Entwicklungsarbeit ist. Durch eine Zusatzplatine, die in den Computer eingebaut werden kann, ist der Amiga 500 mit einer batteriegepufferten Echtzeituhr und weiteren 0,5 MByte erweiterbar. Als Schnittstellen finden sich bei diesem Amiga, der mit einem externen Netzteil ausgerüstet ist, die vom Amiga 1000 her bekannten Standard-Schnittstellen. Auch die bei der seriellen und parallelen Schnittstelle nicht normgerecht belegten Pins wurden jetzt dem Standard angepaßt.

Beim Amiga mußte bislang mit zwei Disketten gebootet werden. Auf der Kickstart-Diskette waren die residenten Gerätetreiber und das AmigaDOS untergebracht. Diese hardwarenahen Teile der Systemsoftware sind jetzt in einem 1 MBit-ROM enthalten. Nur noch die Workbench, die die grafische Benutzeroberfläche Intuition zur Verfügung stellt, muß beim Booten geladen werden.

Während der Amiga 500 durch Steckkarten nicht intern erweitert werden kann, bietet der Amiga 2000 gleich drei Arten von Steckplätzen. Neben dem 86poligen MMU-Slot, der für Speichererweiterungen und Coprozessoren gedacht ist, gibt es im Amiga 2000 noch 100polige Amiga-Slots und 62 polige PC/AT-Slots. Dieses offene System wird von Commodore in drei Grundversionen angeboten, die nach dem Baukastenprinzip beliebig ausgebaut werden können. Der preiswerteste Vertreter der neuen Amiga-Generation mit 1 MByte RAM, zwei 3½-Zoll-Floppy-Laufwerken, RGB-Monitor und Maus soll deutlich unter 4000 DM kosten. In da robuste Stahlblechgehäuse können zwe 3½-Zoll-Floppy-Laufwerke, ein 5¼-Zo Floppy-Laufwerk, oder eine 3½-Zoll- be ziehungsweise 5¼-Zoll-Festplatte einge baut werden. Der weitere Ausbau de Computers durch externe Laufwerke ode durch Festplatten-Steckkarten ist vorge sehen. Als Tastatur dient dem Amig 2000 eine vorbildliche, abgesetzte Einhe mit 96 Tasten.

Commodore ist es mit dem Amiga 200 gelungen, die Leistungsfähigkeit des Ami ga mit der Flexibilität der offenen System wie Apple-II oder IBM-PC/XT zu verbin den. Wer mit dem Amiga gleichzeitig ei nen PC-kompatiblen Computer betreibe möchte, braucht jetzt nicht mehr auf di externe Sidecar-Lösung zurückzugreifer sondern kann eine PC-Emulator-Kart einsetzen, die als Steckkarte realisier wurde. Einfach in den Amiga einstecker und schon ist der Computer gleichzeitig ein MS-DOS- und ein Amiga-Computer Hier liegt die Betonung eindeutig au gleichzeitig, denn der Amiga kennt ja be kanntlich echtes Multitasking. Das MS DOS ist nur eine der zahlreichen Tasks die der Amiga ausführen kann. Alternati zum MS-DOS ist auch der Einsatz vo CP/M-86, Concurrent CP/M oder -DOS sowie Unix möglich. Durch die PC-Emula tor-Karte wird der Amiga befähigt, PC Steckkarten zu betreiben. Das gesamt Spektrum an PC-Erweiterungskarte steht jetzt dem Amiga zur Verfügung, vo der EGA-Karte bis zur I/O-Karte für tech nische Anwendungen. Der 68000er de Amiga kommuniziert mit dem 8088 de PC-Emulators über ein Dual-Port-RAM das 128 KByte umfasst. Über einen 6 KByte großen Teilbereich findet der Datei Transfer statt. Die Bildschirminformatio nen der PC-Karte werden in einem 8 KBy te (monochrom) oder 32 KByte (colo großen Bereich an den Amiga übergeben Der PC im Amiga funktioniert wie ei selbstständiger PC und schreibt seine In formationen in das Dual-Port-RAM, vo wo aus sie vom Amiga geholt und weite verarbeitet werden. Eine Festplatte kann von beiden Betriebssystemen aus genutz werden, wobei der Amiga bis zu 16 GByte adressieren kann. Die Brücke zwischen den Prozessorfamilien kann natürlich auch von einem 80286 oder einem 80386 genutzt werden. Commodore bezeichne die PC-Brückenkarte im Amiga mit „Ja nus", frei nach dem römischen Gott, de gleichzeitig in zwei Zeiten blicken konnte.

In Bezug auf den Preis für die kleinste Lösung mit der PC-Karte wurde vor de offiziellen Vorstellung auf der CeBit vor

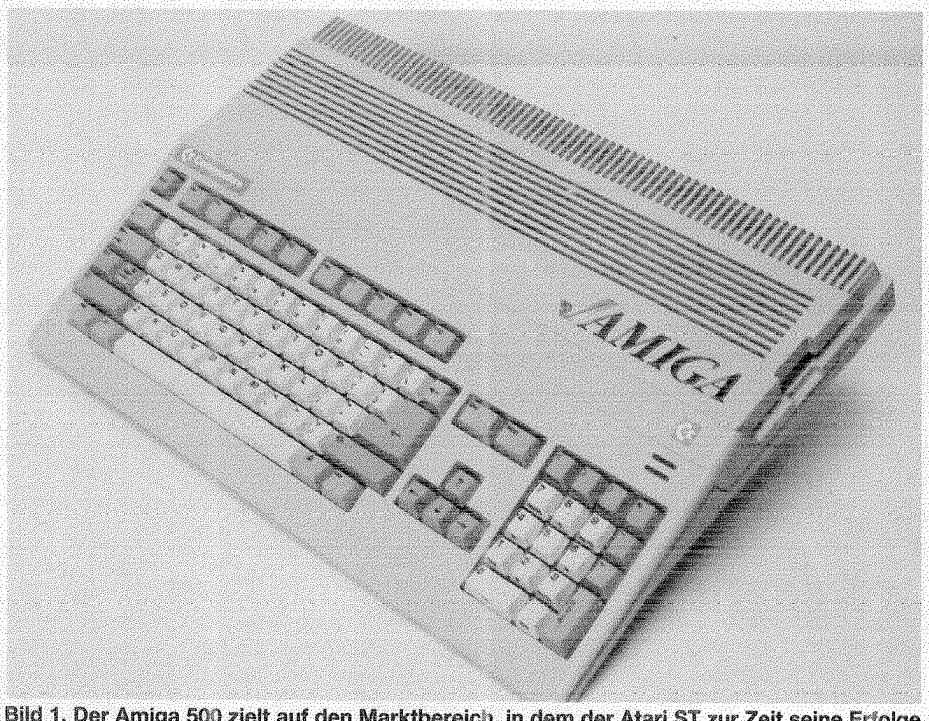


Bild 1. Der Amiga 500 zielt auf den Marktbereich, in dem der Atari ST zur Zeit seine Erfolge feiert – mit einem Preis von knapp 1300 DM ein ernsthafter Konkurrent

Bild 2. Ein professionelles System: Der Amiga 2000

„unter 5000 DM" gesprochen. Damit ist der Amiga 2000 für alle interessant geworden, die etwas mehr als einen einfachen Computer haben wollen; die ein System möchten, mit dem man neben den üblichen Aufgaben für einen PC auch solche aus den Bereichen Musik, bewegte Grafik und Video lösen kann. Und das bei echtem Multitasking. Eine Unix-Steckkarte für den Amiga 2000 wurde für Anfang 1988 angekündigt.

Besonders interessant bei der Vereinigung der zwei Prozessorfamilien in einem Gehäuse ist, daß Daten und Grafiken über einen besonderen Speicherbereich (Clipboard) beliebig zwischen einzelnen Programmen getauscht werden können. Die gewünschten Bereiche werden dabei einfach mit der Maus auf dem Bildschirm ausgewählt.
Der Amiga 2000 wird ab der CeBit ausgeliefert. Die Produktion dieses Computers läuft in Braunschweig schon seit einiger Zeit und soll bis auf 150 Systeme pro Tag gesteigert werden. Von diesem System möchte Commodore etwa 1000 bis 1500 im Monat absetzen. Das kleinere Modell Amiga 500 wird es erst ab Mai/Juni geben. Trotzdem strebt Commodore an, etwa 100.000 davon in diesem Jahr zu verkaufen.

PC-Bus mit EC-Bus verbinden

Die Firma Graf, Kempten, wird auf der CeBit-Messe in Hannover einen Platinensatz vorstellen, der den PC-Bus mit dem EC-Bus verbindet. Diese Verbindung ist nicht nur ein Signalumsetzer, sondern sie ist so konstruiert, daß er PC-XT der Master ist, der nach Belieben in den ECB-Subrechner hineinschauen kann. Dabei sehen die I/O-Ports und der Speicher des ECB-Systems wie Speicheradressen des PCs aus (EXXXXH). Das Konzept ist so gestaltet, daß beide Systeme unter der Anleitung des PCs parallel arbeiten können. Der PC kann auch direkt der Peripherie des ECB-Systems Befehle erteilen. Zum Beispiel kann eine Grafik-Karte direkt und schnell angesteuert werden - oder was es da sonst an Peripherie alles gibt. Besonders interessant ist dieser Brückenschlag für die Besitzer des NDR-Computers, da dieser sehr viele und interessante Peripheriebausteine besitzt. Das Interface besteht aus zwei Platinen, von welchen eine in den ECB-Rechner (oder NDR-Computer) gesteckt wird, während die andere in den PC-Slot gehört.

Victor-Leasing

Die Victor Technologies, Frankfurt, starte im neuen Jahr mit einem Leasing-Programm. Victor PC-Anwender können ab 224 DM monatlicher Leasingrate bei 3 Monaten Laufzeit IBM-kompatible P leasen. Die Benutzer können mit diesen Angebot ihre Liquidität erhöhen und di Steuerlast verringern, denn Leasingrate sind Betriebsausgaben, die sofort und i voller Höhe steuerlich abzusetzen sind.

Macintosh 15 % billiger

Die Apple Computer GmbH, München hat die Preise für den Macintosh Plus u 15 Prozent gesenkt. Ab Februar kostet di Standardversion in Deutschland 847 DM. Damit sollen die durch den Dollarkur bedingten guten Einkaufspreise weiterge geben werden. Gleichzeitig mit der Preis senkung startet Apple eine Sonderaktio mit zwei Macintosh-Paketen. Mit Fest platte kostet der Macintosh 12 595 DM eine Desktop Publishing-Einheit mi Macintosh Plus, Festplatte, Laserdrucke Plus, MacWrite und MacDraw soll 29 95 DM und damit 6759 DM weniger als di Einzelkomponenten kosten.

GfA-Referenzkarte

Ab sofort ist eine kostenlose Referenzkar te zum GfA-Basic für den Atari ST erhält lich. Die praktische Karte enthält die Be fehle des Interpreters übersichtlich nac Gruppen geordnet. Jeder registrierte Be sitzer des GfA-Interpreters, der der sein Adresse und die Seriennummer seines In terpreters angibt, bekommt die Referenz karte gegen Einsendung eines frankierter C5-Rückumschlages von GfA-System technik.

GEM-Hotline

Digital Research hat eine GEM-Hotlin eingerichtet, um registrierte Kunden be allen Fragen zu GEM-Software direkt be raten zu können. Der Hotline Support is für 30 Tage, gerechnet ab der Registrie rung, kostenlos. Danach wird für den Ser vice eine Jahresgebühr von 99 DM erho ben. Zu erreichen ist der Service derzei unter der Telefonnummer 089/5706734 und zwar Mittwochs und Freitags u 14.00...17.30 Uhr.

Dynamisches Datei-System für Modula-2

Ein dynamisches Datei-System, das eine Erweiterung des Modula-2/86-Compilers von Logitech darstellt, bietet jetzt die schweizerische Firma Terra Datentechnik, Zürich, an. Mit Multikey-ISAM steht Entwicklern datenintensiver Anwenderprogramme ein leistungsfähiges Hilfsmittel für einen vereinfachten Zugriff über vordefinierte Zugriffskriterien zur Verfügung. Es erlaubt den Datenzugriff über mehrere vordefinierte Schlüssel. Bei den heute üblichen Singlekey-ISAM-Systemen ist das nur mit erheblichem Mehraufwand möglich.

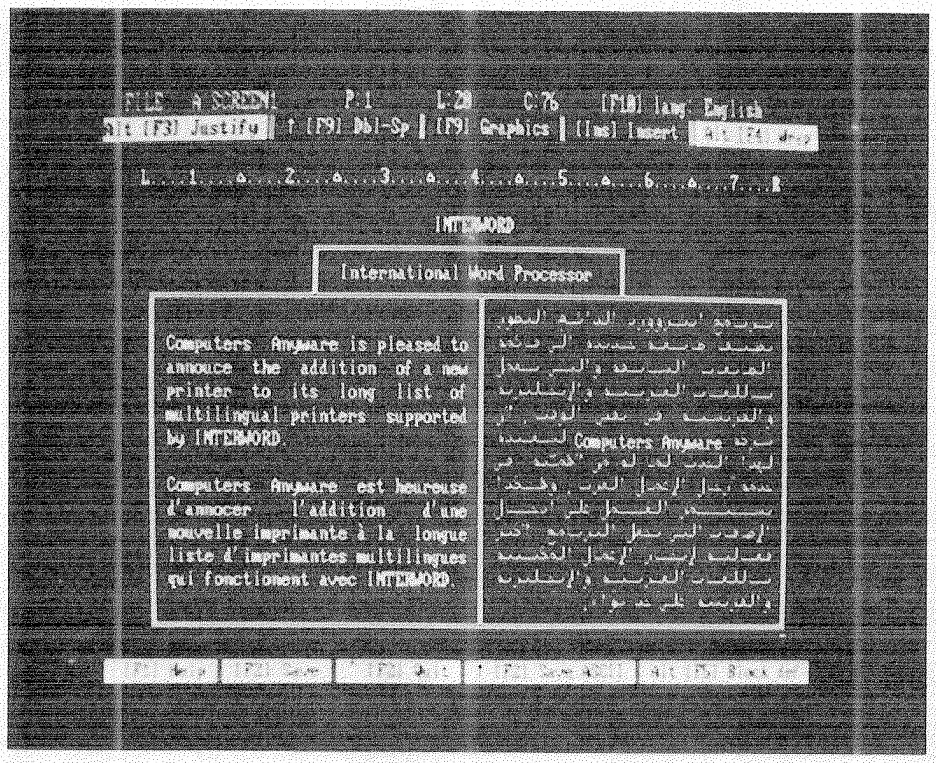
Das Programm läuft auf allen MS-DOS-Rechnern. Es erlaubt eine unbeschränkte Anzahl von Records pro Datei und eine maximale Recordgröße von 65534 Byte. Dateilängen von maximal 4 GByte werden unterstützt. Möglich sind 30 Zugriffsschlüssel mit einer Länge von je 50 Zeichen.

Textverarbeitung mehrsprachig

Wer mit seinem PC Schriftstücke in fremden Sprachen erstellen will, kennt das Problem: Die üblichen Textverarbeitungsprogramme sind jeweils nur für eine Sprache ausgelegt. Mit ihrer Interword-Serie will die amerikanische Computers Anyware Inc. diesem Zustand jetzt ein Ende bereiten. Bei diesen Programmen ist es möglich, per Tastendruck von einer Sprache auf eine andere umzuschalten. Zusätzlich erlauben sie einfache Strichgrafiken.

Mit Latin-Alphabet-Interword lassen sich Texte in Englisch, Deutsch, Französisch und Spanisch bearbeiten. Multilingual-Interword kennt elf Sprachen: die bereits genannten und zusätzlich Dänisch, Finnisch, Isländisch, Norwegisch, Portugiesisch und Schwedisch. Neben anderen Versionen ist auch eine für Arabisch erhältlich.

An Druckern werden Laser-, Typenrad- und Matrixgeräte unterstützt. Optimal ausgenutzt werden die Eigenschaften der neuen Textverarbeitungsprogramme, wenn man Qume-kompatible Typenraddrucker oder Matrixdrucker mit ladbaren Zeichensätzen verwendet.



Schriftstücke in mehreren Sprachen auf dem Bildschirm: für die Textverarbeitungsprogramme der Interword-Serie kein Problem

Modula-2

Software-Entwicklungssystem

M2SDS, das Modula-2 Software-Entwicklungssystem bietet Ihnen:

- ein ausführliches deutsches Handbuch.
- volle 8087-Unterstützung, nicht nur Emulation.
- einen Windowmanager, mit dem Sie gleichzeitig mehrere Programme editieren und ansehen können, mit Uhr, Rechner und ASCII-Tabelle.
- komfortabler Bibliotheksmanager, sorgt für geringen Speicherverbrauch und hohen Komfort auch auf Diskettensystemen.
- syntaxgesteuerter Editor, das Beste, um Modula-2 zu lernen, beschleunigt den Editiervorgang ums 2-3-fache und lässt praktisch keine Tippfehler mehr zu.
- inkrementeller, superschneller Compiler.
- Linker für beliebig grosse Programme; erzeugt direkt ausführbare EXE-Programme.
- alle 20 Module werden im Sourcecode geliefert.

M2SDS ist das modernste und komfortabelste Software-Entwicklungssystem auf dem PC-Markt und kostet nur
DM 299.90 +MWSt./SFr. 267.50

Als Erweiterungen stehen die folgenden Bibliotheken und Programme zur Verfügung:

Erweiterte Bibliotheken **DM 166.60 +MWSt./SFr. 147.–**
Enthält eine ganze Reihe von nützlichen, leistungsfähigen Modulen, z.B. mehrere Implementationen zur Handhabung der RS232-Schnittstelle (Poll-, paralleler Prozess), LONGSET für über 2000 Setelemente, usw.

FOMI **DM 166.60 +MWSt./SFr. 147.–**
Der Foreign Object Module Importer importiert Assemblerprogramme in Modula-2-Module. Diese können dann wie ganz gewöhnliche Module aufgerufen und gelinkt werden.

M2MAKE **DM 336.60 +MWSt./SFr. 297.–**
Erkennt alle Abhängigkeiten zwischen Modulen und compiliert bei einer Veränderung der Abhängigkeiten oder wenn unterschiedliche Versionen vorliegen genau die betroffenen Module automatisch neu. Ein Muss für alle professionellen Programmierer, da es sehr viel Zeit spart.

SDS-XP **DM 900.– +MWSt./SFr. 750.–**
Enhält M2SDS, erweiterte Bibliotheken, FOMI und M2MAKE.

P2M-Converter **DM 320.– +MWSt./SFr. 280.–**
Konvertiert Turbo-Pascal-Programme in syntaktisch korrekte Modula-2-Programme. Übertragungsgeschwindigkeit: ca. 1000 Zeilen/min, Konvertierungsgrad: 95 – 100%, diverse Module im Sourcecode zur Unterstützung der Konvertierung (Turbo – enthält alle speziellen Fähigkeiten von Turbo-Pascal*, SetOps – stellt Pascal-Setopperationen auch in Modula-2 zur Verfügung usw.

M2DEBUG **DM 506.60 +MWSt./SFr. 447.–**
Symbolisch, interaktiver Debugger mit während des Debuggens aufrufbaren Hilfetexten. Erlaubt sogar das Debuggen von systemnahen und auch parallelen Prozessen.

ROM-Version von M2SDS
DM 750.– +MWSt./SFr. 600.–

* M2SDS ist ein Warenzeichen von Interface Technologies, Turbo-Pascal von Borland.

Bezugsquellen:

Bundesrepublik Deutschland:
- Interplan, bei der Pfizbuche 77, 7900 Ulm, 0731/2 69 49
- E. Jurschitza, Eisenindstr. 7a, 8900 Augsburg, 0821/8 57 37
- SW-Datentechnik, Raiffeisenstr. 4, 2085 Quickborn, 04106/39 98
- Wilken, Ratsbleiche 1, 3300 Braunschweig, 0531/34 72 75
- Lauer & Wallwitz, Erkönigweg 9, 6200 Wiesbaden, 06121/4 27 71

Schweiz:
- Frei-Elektronik, Stationsstr. 37, 8604 Volketswil, 01/945 54 32

Österreich:
- ICA GmbH, Heigerleinstr. 9, 1160 Wien, 0222/454 50 10

oder bei Ihrem nächsten Computer- oder Buchhändler.

Generalvertrieb für Europa:

A. + L. Meier-Vogt
Im Späten 23
CH-8906 Bonstetten/ZH
Tel. (41) (1) 700 30 37

Superschneller Multiplizierer

In den Labors der kanadischen Firma Bell-Northern Research (BNR) ist ein Gallium-Arsenid-Baustein entwickelt worden, der zwei vierstellige Dualzahlen innerhalb einer Nanosekunde multipliziert. Kommerziell erhältliche Silizium-Chips arbeiten rund fünf- bis zehnmal langsamer.

Derart schnelle Multiplizierer (und andere Elemente) braucht man z. B. für die digitale Signalverarbeitung in künftigen Kommunikationseinrichtungen. Mit Silizium als Basismaterial, wie es heute in den meisten ICs zu finden ist, kann man solche Geschwindigkeiten kaum mehr realisieren. Seit vielen Jahren experimentiert man deshalb mit Gallium-Arsenid-Verbindungen, die sich bis jetzt allerdings nicht auf breiter Front durchsetzen konnten.

Neben dem Material spielen die Abmessungen der Stukturen auf dem Chip eine wesentliche Rolle für die erreichbare Geschwindigkeit. Bei dem von BNR entwikkelten Multiplizierer sind die Verbindungen zwischen den einzelnen Elementen auf dem Chip nur noch bis zu 1 µm breit, das ist nur wenig mehr als die Wellenlänge des sichtbaren Lichts. Trotzdem beherrscht man die Herstellung solcher Strukturen heute schon in der Massenfertigung. Der Multiplizierer hat also berechtigte Aussichten, das Laborstadium bald zu verlassen.

Bargeldlos telefonieren

In einem Großversuch testet die Deutsche Bundespost derzeit öffentliche Fernsprecher, die nicht mehr mit barer Münze bezahlt werden, sondern mit einer Art Kreditkarte. Seit Dezember sind 200 dieser Apparate in Flughäfen und Hauptbahnhöfen von 17 deutschen Städten im Einsatz. Hersteller ist die Firma Siemens.
Auf den Buchungskarten, die im Format gängiger Scheckkarten gehalten sind, befindet sich ein Speicher-Chip. Dieser Chip enthält Informationen, die von der Telefonelektronik ausgewertet werden. Zwei Arten von Karten sind erhältlich: Bei der einen zahlt man im voraus, und das Restguthaben wird jeweils vom Telefonapparat angezeigt. Bei der anderen werden die Gebühren direkt vom Fernmeldekonto abgebucht. Im nächsten Jahr will die Bundespost damit beginnen, Kartentelefone überall als Ergänzung zu den heutigen Münzfernsprechern einzusetzen. Damit wird in Zukunft das Aufbrechen des Münzspeichers sinnlos.

Das Suchen nach dem „Telefonzehnerl" entfällt: Bezahlt wird mit der Buchungskarte

Modula-2

Software-Entwicklungssystem

Neben unseren Implementationen von Modula-2 Entwicklungssystemen auf verschiedenen Computern und von Farsight, der BESONDEREN Software bieten wir Ihnen als Software-Verlag eine ganze Reihe von Toolboxen und Werkzeugen an, um Ihr Leben als Modula-2-Programmierer zu verschönern.

S-PRINT **DM 75.– +MWSt./SFr. 60.–**
Macht jegliche Druckeranpassung überflüssig. Jeder Drucker ist mit jedem Programm beliebig über S-PRINT konfigurierbar. Dieses Werkzeug macht manche Seitenbeschreibungssprache überflüssig. Vom Telex bis zum Laserdrucker anwendbar.

M2Graph **DM 95.– +MWSt./SFr. 80.–***
Stellt sämtliche Standardfunktionen zur Herkuleskarte zusammen mit anschaulichen Beispielprogrammen bereit.

RTA Utilities Disk **DM 95.– +MWSt./SFr. 80.–***
Beschleunigt die Datenein-/ausgabe um das 2-10-fache, erweitert die mathematischen Funktionen um inverse trigonometrische und hyperbolische Funktionen, erlaubt Logarithmen zu jeder Basis, enthält REAL/INTEGER – INTEGER/REAL Konversion und stellt komfortable Ausgaberoutinen für REALs bereit.

Window-Management-System
DM 149.– +MWSt./SFr. 139.–*
Erlaubt komfortable und modernste Handhabung von beliebig vielen Fenstern (neudeutsch: Windows). Ausserordentlich hohe Geschwindigkeit, sehr flexibel (Farben, Attribute, Grösse, Position, Scrollen, usw.). Enthält als Zugabe einen umfangreichen Menuhandler.

Adressverwaltung **DM 175.– +MWSt./SFr. 149.–***
Ohne Zweifel eine der schnellsten und komfortabelsten Adressverwaltungen. 10 Selektionskriterien pro Adresse, 12 frei belegbare Textzeilen, Sortierung alphabetisch und nach Postleitzahlen. Das beste Beispielprogramm zur Demonstration der Effizienz von Modula-2. Beliebig anpassbar, da Sourcecode mitgeliefert wird. (Die Adressverwaltung baut auf M2SDS, B-Tree-ISAM und Power M²/FORM auf.)

* wird mit Sourcecode geliefert.

Ausserdem führen wir ein riesiges Sortiment an Literatur und Büchern zu Modula-2, sowie u.a. die folgenden Werkzeuge und Bibliotheken:

- **POWER M², /FORM, /STAT, /TECH**
- **Hobson's Choice**
- **B-Tree-ISAM**
- **GEM-Bindings für Modula-2**
- **GKS Graphic Kernel System**

Dank Modula-2 ist es uns möglich, all diese qualitativ hochwertigen Zusätze und Hilfsmittel so günstig verfügbar zu machen. Sie wissen ja selbst, dass Modula-2 standardisiert ist. Somit müssen alle Werkzeuge nur einmal geschrieben werden und stehen dann auf praktisch allen Rechnern zur Verfügung.

Als Software-Verlag unterstützen wir unsere Anwender nicht nur mit einem erstklassigen Informations- und Update-Service, sondern auch, indem wir ständig auf der Suche nach neuen Modula-2-Anwendungen und -Produkten sind, die wir auf dem internationalen Markt allen Anwendern verfügbar machen. Modula-2-Programmierer erhalten von uns Unterstützung bei der Erstellung, Dokumentation und internationalen Vermarktung ihrer Produkte auf den verschiedensten Computertypen.

Wenn Sie an Modula-2 denken, denken Sie an uns.

FARSIGHT ist ein Warenzeichen von Interface Technologies Corp.

Bezugsquellen:

Bundesrepublik Deutschland:
– Interplan, bei der Pilzbuche 77, 7900 Ulm, 0731/2 69 49
– E. Jurschitza, Elfenstraße 7a, 8900 Augsburg, 0821/8 57 37
– SW-Datentechnik, Reiffeisenstr. 4, 2085 Quickborn, 04106/39 98
– Wilken, Ratsbleiche 1, 3300 Braunschweig, 0531/34 72 75
– Lauer & Wallwitz, Erlkönigweg 9, 6200 Wiesbaden, 06121/4 27 71

Schweiz:
– Frei Elektronik, Stationsstr. 37, 8604 Volketswil, 01/945 54 32

Österreich:
– ICA GmbH, Heigerleinstr. 9, 1160 Wien, 0222/454 60 10

oder bei Ihrem nächsten Computer- oder Buchhändler.

Generalvertrieb für Europa:

A. + L. Meier-Vogt
Im Späten 23
CH-8906 Bonstetten/ZH
Tel. (41) (1) 700 30 37

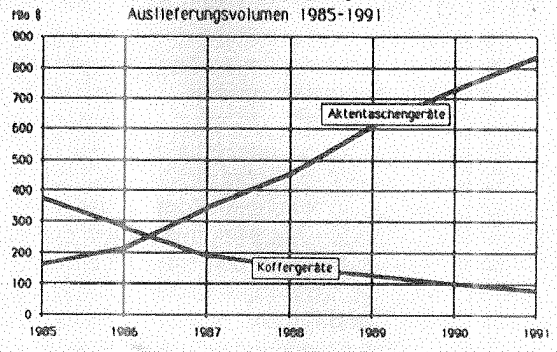
Marktstudie: Tragbare Computer

Mit dem westeuropäischen Markt für tragbare Computer in den Jahren 1985 bis 1990 befaßt sich eine Studie der IDC Deutschland GmbH. Dabei unterscheiden die Forscher zwischen Koffergeräten (Portables) und Aktentaschengeräten (Laptops). Die Systeme der ersten Kategorie benutzen durchweg eine herkömmliche Elekronenstrahlröhre zur Anzeige und wiegen deutlich über 8 kg, während „Laptops" mit LC- oder Plasma-Display ausgestattet sind und etwa 5 kg wiegen. 1986 wurden nach Einschätzung von IDC europaweit insgesamt 265 500 Geräte verkauft. Das entspricht einem Wert von 496 Mio. $ und einem Anteil von rund 9 % am gesamten PC-Markt. Nach Stückzahlen gerechnet hielten sich Koffer- und Aktentaschencomputer etwa die Waage, mit leichten Vorteilen für die größeren Geräte. Das soll sich aber drastisch ändern: 1991 traut man den Aktentaschencomputern 1 Mio. Stück (835 Mio. $) zu und den „Portables" nur noch 70 000 (74 Mio. $). Diese rasante Entwicklung wird auf der Anbieterseite zu einem harten Konkurrenzkampf führen. Wie das ablaufen kann, zeigt der Vergleich der führenden Anbieter in den Jahren 1984, 1985 und 1986 (bis Jahresmitte). 1984 war Epson mit 47 % unumstrittener Marktführer vor Olivetti mit 17 %. Ein Jahr danach schrumpfte Epsons Anteil auf 20 %. Trotzdem blieb das Unternehmen an der Spitze, vor Ericsson (18 %) und Toshiba (13 %). Olivetti hatte sich von diesem Markt verabschiedet. Mitte 1986 war Toshiba souveräner Spitzenreiter mit 33 % und Epson nahm mit 17 % Platz zwei ein.

AKTENTASCHEN LASSEN KOFFER LIEGEN

Der europäische Markt für tragbare Personal Computer
Auslieferungsvolumen 1985-1991

Dem unaufhaltsamen Aufstieg der Aktentaschencomputer steht der stetige Abstieg der Koffercomputer gegenüber



	Aktentaschengeräte	Koffergeräte
1986	129.800 Stk, 214 Mio $	135.700 Stk, 282 Mio $
1991	990.000 Stk, 835 Mio $	66.760 Stk, 074 Mio $

Begraben Sie Ihren

SCHNITTSTELLENTESTER...

...jetzt gibt es »COMIX«

»COMIX«: Die reine Softwarelösung zur Darstellung, Analyse und Steuerung von RS232/V.24-Schnittstellen nur über den Monitor!

»COMIX«: Ein deutsches Produkt der ComFood Software GmbH mit 50seitigem Handbuch!

»COMIX«: Support direkt vom Hersteller!

»COMIX«: Endkundenpreis: **299,— DM**

Fordern Sie gleich ein Datenblatt zu dieser einmaligen Lösung an!
Händleranfragen willkommen.

ComFood Software GmbH, Rohrbusch 79, 4400 Münster-Roxel, Tel. 0 25 34/70 93

CeBit, vertreten bei Phoenix Technologies, Halle 6, Stand C46-F23

– Wir füttern Ihren Computer –

ComFood Software GmbH

Der mc-68000-Computer in neuer Form

Nur wenig ist bekanntlich so gut, daß man es nicht mehr verbessern kann – das gilt auch für den mc-68000-Computer. Er erfreute sich von Anfang an großer Nachfrage, mittlerweile sind einige tausend davon in Betrieb. Im Laufe der Zeit ergaben sich verschiedene Verbesserungsvorschläge, die jetzt in einem neuen Leiterplatten-Design vereint wurden.

Die wichtigste Verbesserung besteht darin, daß der Rechner nun auf einer sechslagigen Multilayer-Platine aufgebaut ist. Damit fallen ein für alle Mal sämtliche Probleme mit Stromversorgung und Störeinstrahlungen weg. Mit ihrer hohen Zuverlässigkeit und Unempfindlichkeit gegenüber Störfeldern eignet sich die neue Platine uneingeschränkt für den professionellen Einsatz.

Trotzdem ist der neue mc-68000-Computer voll kompatibel zum Vorgänger. So lassen sich alle Eweiterungskarten und sämtliche Software weiterverwenden. Unterprogramm-Einsprünge der Monitor-EPROMs und Hardware-Adressen wurden unverändert beibehalten. Und obwohl zahlreiche Verbesserungen und Schaltungserweiterungen vorgenommen wurden, reicht diese Kompatibilität bis zu den Abmessungen: Die neue Karte paßt in die selben Bohrlöcher und verlangt die gleichen Gehäuse-Aussparungen. Dank der Multilayer-Technik wurde dabei aber die Bauteile-Dichte gegenüber der ersten Version des mc-68000-Computers um einiges erhöht.

Neben der Korrektur kleiner Schönheitsfehler gab es auch einige recht weitreichende Änderungen. So wurde die Signaltrennung von Bus und systeminternen Signalen stark verbessert. Der aktive Busabschluß befindet sich nun gleich auf der Hauptplatine; Floppykontroller und SASI-Schnittstelle sind ebenfalls direkt auf der Hauptplatine untergebracht. Ein Hybrid-Quarzoszillator garantiert eine optimale Takterzeugung. Die V.24-Schnittstelle hat nun einen eigenen Kommunikationsbaustein (EPCI 2661, 19200 Baud, vollduplex, Spooler), der sich ebenfalls – wie auch die Tastaturschnittstelle für IBM-PC-Tastatur – bereits auf der Hauptplatine befindet.

Der Prozessor 68000 kommt im platzsparenden PIN-GRID-Gehäuse zum Einsatz. Das Timing im RAM- und Videobereich wurde verbessert und ein zusätzlicher „DAISY-Chain" auf dem Bus für Interrupt und DMA vorgesehen.

Veränderungen wurden auch am Userport vorgenommen: Statt eines 50poligen Steckers gibt es nun einen 40poligen für den eigentlichen Userport und einen 26poligen Stecker für den Abgriff der Video-Signale. Am Video-Ausgang lassen sich alle Synchron-Signale mit normalem und inversem Pegel, das Bildsignal in Composite- und TTL-Pegel sowie die RGB- und Intensity-Signale abgreifen. Beide Steckverbinder sind mit „twisted-pair"-Masse-Leitungen versehen, so daß sich auch längere Flachkabel anschließen lassen.

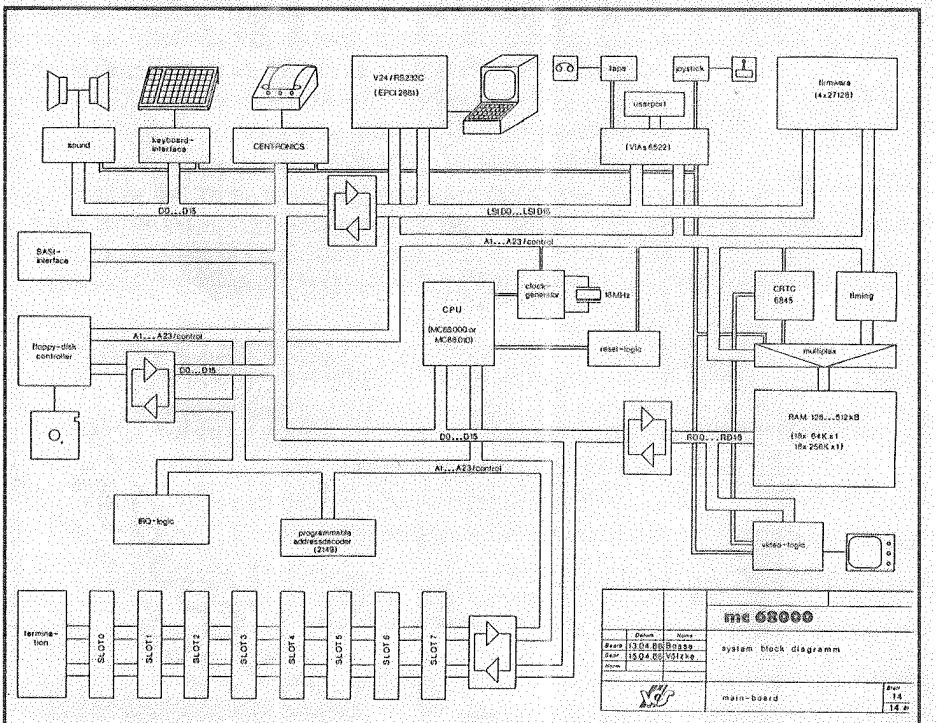
Die Logik zur Erzeugung eines FBAS-Signals wurde nicht mehr integriert. Dafür wurde das RGB-Signal (320 x 200 Punkte) auf maximal 16 Farben erweitert (bei Verwendung eines RGBI-Monitors). Per Software kann der RGBI-Ausgang auch auf grün/schwarz-Betrieb mit 80 Zeichen (640 x 200 Punkte) pro Zeile umgeschaltet werden.

Durch die Integration dreier wichtiger Erweiterungskarten des alten Systems (Floppykontroller, Busterminierung und SASI-Schnittstelle) auf der Multilayer-Hauptplatine stehen zum einen 3 Steckplätze mehr zur Verfügung, außerdem sind nun wirklich alle zum Disketten-Betrieb nötigen Schnittstellen bereits auf der Platine enthalten, so daß der neue mc-68000-Computer als Einplatinen-Computersystem ohne zusätzliche Erweiterungskarten betrieben werden kann.

Und natürlich gelten weiterhin die übrigen technischen Daten des mc-68000-Computers: Prozessor MC68000 oder MC68010 mit 8 MHz Taktfrequenz, 512 KByte RAM auf der Hauptplatine, über die Steckplätze bis auf 15 MByte ausbaubar (mit RAM-Karte: 2 MByte pro Steckplatz!), 64 KByte EPROM für System-Monitor und CP/M-68k oder OS-9/68k-Bootstrap, VME-Bus-ähnlicher Systembus, voll gepuffert, aktiv terminiert, auf 8 Steckplätze herausgeführt, zahlreiche Erweiterungskarten lieferbar, programmierbarer Adreßdecoder, Floppy-Schnittstelle für bis zu 8 Laufwerke, SASI-Schnittstelle für Festplatte, Centronics- und V.24-Schnittstelle.

Als Tastatur läßt sich auch weiterhin wahlweise eine IBM-PC- oder kompatible, eine parallele oder (über V.24) eine serielle Tastatur anschließen. Ebenfalls über V.24 ist auch ein Terminal-Betrieb möglich.

Und da alles zum Vorgänger kompatibel bleibt, findet auch die gleiche Software Verwendung: Das 16-KByte-Monitor-Betriebssystem mit Bildschirm-Editor, symbolischem Assembler, Debugger und Monitorprogramm, das mit der Platine ausgeliefert wird, optional 32 KByte für CP/M-68k oder OS-9/68k als Disketten-Betriebssytem. Der neue mc-68000-Computer ist beim Hersteller (DSM Digital-Service GmbH) ab Lager lieferbar.



Das Prinzip ist das gleiche geblieben: Mit dem mc-68000-Computer erhält der Anwender ein offenes System mit einem Maximum an Flexibilität

TEDAS mit Epson ist zuverlässig

Im Juni 1983 wurde unser Teledaten-Service, kurz TEDAS genannt, aus der Taufe gehoben. Damals eine echte Sensation für Datenfernübertragungs-Freaks, denn TEDAS war der erste nicht nur Insidern bekannte Dienst, der einen Computer präsentierte, der von jedermann angerufen werden konnte. Die Sache funktionierte und funktioniert so: man benötigte (und benötigt heute auch) im einfachsten Fall einen Akustik-Koppler, der die zu übertragenden Bytes zur Fernübertragung mit 300 Baud über das öffentliche Fernsprechnetz aufbereitet und etwas Software für den Computer, damit er voll duplex mit 300 Baud über seine V.24-Schnittstelle und das Modem mit unserem Teledaten-Service-Computer sprechen kann. Ganz genau gesagt: Das benötigte Terminal-Programm muß auf 300 Baud, 8 Datenbits, keine Parity und 1 Stopbit eingestellt werden. Dann kann die Nummer (0 89) 59 64 22 (oder zweiter Rechner: 59 84 23) angerufen werden. Der TEDAS-Rechner sendet bei durchgeschalteter Verbindung seinen Answer-Ton zum Anrufer und wartet auf dessen Echo, mit anderen Worten, er wartet darauf, daß Sie Ihren Telefonhörer auf den Akustik-Koppler pressen. Danach bekommt er dessen Originate-Ton zu hören und antwortet nun mit der Aufforderung „bitte Return drükken". Damit überprüft er, ob eine Verständigung möglich ist. Wenn nicht, dann bricht er die Kommunikation mit der Bemerkung „keine Verständigung möglich" ab. Wenn das Return-Zeichen durchkommt, führt er Sie durch ein selbsterklärendes Menü, das Ihnen unser Angebot präsentiert. Dabei wird auch die Zahl der Anrufer präsentiert, die in den vergangenen drei Jahren angerufen haben. Es sind momentan über 150 000 auf der ersten Nummer und über 103 000 auf der zweiten Nummer, die später installiert wurde. Was wird geboten? Es sei nur erwähnt, daß wir Firmenadressen anbieten, Programme aus dem Heft, Vorinformationen zum Inhalt des neuesten Heftes, Neues aus dem Software-Service, daß wir eine kleine Mailbox-Ecke zu Fragen an die Redaktion eingerichtet haben und eine Art pin-board, in dem Sie alles mögliche an Angeboten, Fragen und Antworten von TEDAS-Anrufern lesen können. Rufen Sie doch mal an! Ein interessantes Detail zum Schluß: Die Rechner hinter den TEDAS-Anschlüssen sind ganz gewöhnliche CP/M-Rechner. Nach dem, was sie bisher geleistet haben, sind sie aber doch auch äußerst ungewöhnlich. Denn bei jedem Anruf wird nicht nur automatisch abgehoben, sondern bei gelungenem Kontakt wird auf das Floppy-Laufwerk zugegriffen und von dort werden die Zahl der Anrufer und die angeforderten Dialogteile geholt. Dies ist nun ein echter Dauertest der Rechner, wobei einer der beiden seit drei Jahren ununterbrochen Dienst tut. Dabei waren seine Floppy-Laufwerke nach etwa 50 000 Anrufen verschlissen und wurden ausgewechselt. Etwa einmal im Jahr geben die von uns verwendeten Floppy-Scheiben den Geist auf, es ist dann mehr als 30 000 mal auf die Direktory-Spur zugegriffen worden. Die Computer, die so klaglos laufen, das sind zwei QX 10 von Epson, die uns damals geeignet erschienen, weil sie für uns durchsichtige Kommunikationsroutinen im BIOS und – für den offiziellen Betrieb wichtig – auch den Segen der Post in Form einer amtlichen Zulassung besaßen.

Zeitschrift auf Diskette

Der Dr. Alfred Hüthig Verlag aus Heidelberg vertreibt das kanadische Magazin „FaSTer Disk Mag" für alle Atari ST Computer. Das Magazin wird als Diskette geliefert und erscheint ab Januar 1987 alle zwei Monate. Neben Produktbeschreibungen für den Atari ST sind auf der Diskette Desk Accessories, Spiele und Anwenderprogramme enthalten. Der Umfang jeder Ausgabe, die 24,80 DM kostet und über Buchhandlungen und Computershops vertrieben werden soll, liegt zwischen 80 und 100 Zeitschriftenseiten.

CD-ROM an MS-DOS

Microsoft hat Erweiterungen des Betriebssystems MS-DOS zur Unterstützung von CD-ROM-Speichern an Personal Computern fertiggestellt. PCs, die unter MS-DOS 3.1 oder 3.2 arbeiten, können jetzt Daten lesen, die auf einem CD-ROM im „High-Sierra-CD-ROM-File-Format" gespeichert sind. Durch die Erweiterung ist ein Zugriff auf 550 MByte große Dateien möglich, da die Limitierung auf 32 MByte unter MS-DOS aufgehoben wurde. Bestehende MS-DOS-Software soll ohne Probleme mit dem neuen Massenspeichermedium arbeiten können. Verfügbar ist das Produkt, von dem Microsoft den hardware-unabhängigen Teil liefert, vorerst nur für OEM-Kunden.

INDUKTIVE SYSTEME

Die Expert-System-Shells der nächsten Generation.

Einbettbarkeit in bestehende Verfahren, Optimierung eingegebener Regelstrukturen durch Lernalgorithmen, exzellentes Zeitverhalten und Source-Code-Generierung sind die hervorstechenden Merkmale dieser in benutzerfreundlicher Windowtechnik angelegten Tools zur Entwicklung von Expertensystemen auf Unix-Workstations oder dem PC.

Intelligent Code Box

Diese AI-Entwicklungsumgebung erzeugt wahlweise Pascal-, C-, Fortran-, Prolog- oder Lisp-Source-Code und ermöglicht so eine flexible Einbindung von Wissensbasen in unterschiedlichste Umgebungen.

Rulemaster

Über die vom System generierten Fortran- oder C-Sourcen leicht in bestehende Applikationen einbettbar.

ExTran 7

Einfache Umsetzung von Wissensbasen in Fortran-Source-Code.

Mit Hilfe der komfortablen Knowledge Engineering Tools **1st Class** und **SuperExpert** kann der Domänenexperte ohne Programmierkenntnisse aktiv in die Erfassung des Domänenwissens einbezogen werden.

Wir beraten Sie bei der Auswahl der geeigneten Werkzeuge für die Implementation Ihrer Anwendungen der Wissensverarbeitung in den Bereichen Diagnose, Prognose, Configuration oder Simulation.
Fordern Sie detaillierte Informationen an:

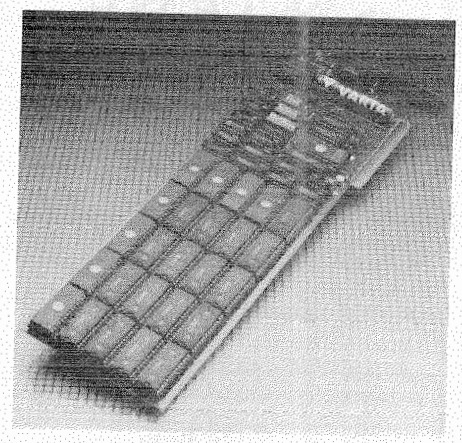
Brainware
Ihr Experte in Expertensystemen
Consulting · Schulung · Software

Brainware GmbH · Voltastraße 5
1000 Berlin 65 · Tel. 030/4 69 46 96
Teletex 308 145-BIG

CeBIT Hannover
4.–11.3.1987
Stand Hewlett Packard
Nr. 712 7k1

Halbleiter-Disk für den PC

Für PCs gibt es jetzt eine Alternative zu den mechanischen Laufwerken – die Silicon-Disk 250 S von Altec-Electronic, Hannover. Durch Halbleiterspeicher können zwei Diskettenlaufwerke nachgebildet werden. Das erste „Laufwerk" ist mit EPROMs bestückt (512...1024 KByte), das zweite mit 544 KByte batteriegepuffertem CMOS-RAM. Neben Programmen und anderen unveränderlichen Dateien läßt sich im EPROM-Bereich auch das Betriebssystem unterbringen. Da der PC von der Silicon-Disk booten kann, ist ein PC ohne mechanisches Laufwerk realisierbar. Interessant ist dies für den Industrieeinsatz unter schwierigen Umgebungsbedingungen.

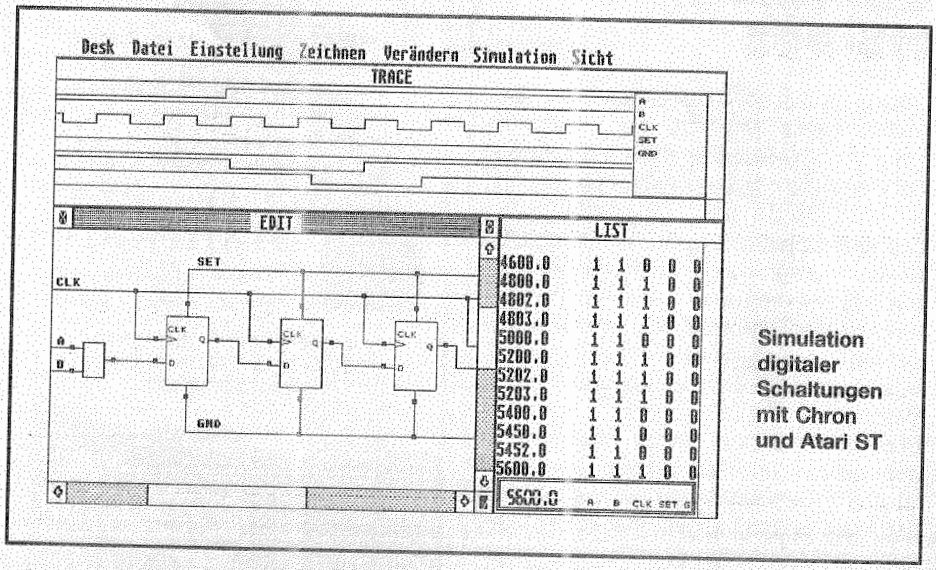


Mit der Silicon-Disk können mechanische Laufwerke ersetzt werden

Durch eine Lithium-Batterie bleiben die Daten des RAM-Laufwerks bis zu fünf Jahren erhalten. Eine Überwachungslogik warnt den Benutzer rechtzeitig vor einem drohenden Datenverlust. Zusätzlich befindet sich auf der Silicon-Disk eine Echtzeituhr mit Kalender. Defekte Speicherbausteine werden durch eine CRC-Prüfung erkannt. Gegenüber herkömmlichen RAM-Disk-Konzepten ist die Silicon-Disk nicht im Arbeitsspeicher, sondern im E/A-Adreßraum des PCs angesiedelt. Es werden acht Portadressen belegt, die mit einem DIP-Schalter einstellt werden können. Die Zugriffszeiten liegen zwischen einem Fünftel und einem Drittel gängiger Festplatten.

Integriertes CAE-Paket für Atari ST

Jetzt gibt es ein leistungsfähiges Werkzeug zur Entwicklung von digitalen Schaltungen und Baugruppen für den Atari ST. Das Programm Chron erlaubt Bauelemente aus einer Bibliothek abzurufen und sie zu einer Schaltung zusammenzusetzen. Dabei ist jedes Bauteil mit grafischen, logischen und zeitlichen Definitionen versehen. Ergänzt durch die Verbindungen der Bauteile untereinander ergibt sich ein Modell der Schaltung, an dem durch Anlegen von Eingangsmustern das Verhalten der Schaltung simuliert werden kann. Ein Worst-Case-Simulator deckt Laufzeitfehler unter Berücksichtigung von Bauteilstreuungen und Layouteinflüssen auf. Durch die konsequente Nutzung von GEM ist das Programm, das auf jedem Atari ST mit monochromem Bildschirm ablauffähig ist, einfach zu bedienen. Bauteile können aus einer Bibliothek entnommen oder vom Benutzer selbst definiert werden. Hardcopies auf Matrixdruckern oder Ausgaben auf Plotter sind ebenso wie Weiterverarbeitung der Simulationsergebnisse mit anderen Programmen vorgesehen. Das Programm kostet etwa 1800 DM. Eine Version, die nur die Eingabe und die Simulation von Schaltungen erlaubt, soll 200 DM kosten. Das CAE-Paket wird von Harald Prillwitz aus München vertrieben.



Simulation digitaler Schaltungen mit Chron und Atari ST

Mailbox für die Bremer

Eine Mailbox betreibt seit November der Club der Sparkasse und LBS Bremen unter der Rufnummer 1 79 20 23. Sie begrüßt den Anrufer mit einem Titelbild und fragt ihn nach dem Namen und dem Paßwort. Wer noch nie mit der Mailbox gearbeitet hat, antwortet beim ersten Anruf mit „Gast". Dann kann er Mitteilungen und Veranstaltungshinweise abrufen oder anderen Anrufern Nachrichten hinterlassen. Eine Kontakt-Ecke gibt unter anderem Auskunft darüber, ob ein anderer Computer-Freak etwas verkaufen will. Die neue Mailbox ist täglich rund um die Uhr in Betrieb.

Nachträge

Turbo-Pascal bedient Hercules-Karte

mc 1987, Heft 2, Seite 46

Im Programm auf Seite 46 ist beim Umbruch eine Zeile verlorengegangen. Bitte fügen Sie nach der 23. Zeile
(PORT[IndexReg] := RegSel;)
noch folgende Anweisung ein:
Port[DataReg]:=GrpVal[RegSel];
Wir bitten den Autor und die Leser um Entschuldigung.

Diskeditor für CP/M-68k

mc 1987, Heft 2, Seite 40

Bei der Entfernung des Programmkopfes wurde auch die Anweisung
#include „stdio.h"
mit entfernt, die (natürlich) am Anfang dieses Programmes nötig ist. Jeder, der schon einmal in C programmiert hat, wird das wohl sofort gemerkt haben.

Hardware für Taschencomputer

Burkhard Kainka, Hardware-Erweiterungen für Sharp-Taschencomputer, Franzis-Verlag, München, 1987, 111 Seiten, 48 DM.
ISBN 3-7723-8351-3

Nach dem Motto „Computer sollen uns nicht nur Arbeit bereiten, sondern auch Arbeit abnehmen" hat sich Burkhard Kainka an die Arbeit zu seinem Buch gemacht, das für jeden Anwender eines Sharp-Taschencomputers geeignet ist. Alle Programme sind für die Modelle PC-1245/51, PC-1260/61, PC-1401/02, PC-1350 und PC-1450 geeignet. Nach einer grundlegenden Einführung in die Maschinenspracheprogrammierung auf den kleinen Computern zeigt der Autor vorbildlich, wie die Ports beschaltet werden. Gleich das erste Beispiel für den sinnvollen Einsatz eines Taschencomputers als Steuergerät liefert B. Kainka mit einer Zeitschaltuhr. Über die Ansteuerung eines Digital/Analog-Wandlers bis hin zum Aufbau eines Frequenzmeßgerätes findet der Leser viele interessante Schaltungen. Bei allen Schaltungen, die durch sehr einfachen Aufbau leicht nachzubauen sind, ist das entsprechende Programm mit einer ausführlichen Erklärung angegeben. Naturgemäß liegt der Schwerpunkt der Ausführungen bei Anwendungen im Bereich der Meß-/Steuer-/Regeltechnik. Wer also zum Beispiel schnelle Vorgänge zu analysieren hat, muß nicht unbedingt einen Transientenrecorder einsetzen – ein Sharp-Taschencomputer mit dem entsprechenden Programm und einer kleinen Zusatzschaltung tut's auch. *rs*

IBM PC AT intern

T. J. Byers, IBM PC AT – Der Rechner der nächsten Generation, 300 Seiten, kart., Verlag McGraw-Hill, Hamburg, 48 DM.
ISBN 3-89028-070-6

Nach den PC-Nachbauten (nicht nur aus Fernost) gibt es jetzt schon eine ganze Reihe AT-Kompatibler. Das Buch von T. J. Byers behandelt das Original, ist aber sicher für alle AT-ähnliche Rechner interessant. In den ersten drei Kapiteln widmet sich der Autor der Hardware, so zum Beispiel den Schalterstellungen auf Hauptplatine und Speichererweiterung, der Systeminstallation und der Tastatur. Anschließend werden DOS 3.0 und 3.1 und die verschiedenen Laufwerkstypen für Disketten und Hard-Disks behandelt. Nach einem Kapitel über die unterschiedlichen Bildschirmkarten wird das Programm Top-View besprochen, das in den USA sicher weit verbreitet ist, in der deutschen Übersetzung hätte das Kapitel sicher wegbleiben können. Nach einem Abriß über die Schnittstellen wird noch auf das Betriebssystem XENIX und die Installation des IBM-Netzwerks eingegangen. Zum Schluß wird auf Kompatibilitätsprobleme mit dem PC, die Uhr und das Basic eingegangen. Die Übersetzung ist bis auf einige Ausrutscher im Kapitel zur „Realzeituhr" ordentlich, das Buch ist für den technisch interessierten AT-Besitzer sicher von Nutzen. *JPL*

PC und lokale Rechnernetzwerke

F. J. Kauffels, Personal Computer und lokale Rechnernetzwerke, Markt & Technik Verlag AG, Haar, 1986, 250 Seiten, 78 DM.
ISBN 3-89090-176-X

Das Buch von Dr. rer. nat. F. J. Kauffels zu PC-LAN gliedert sich in drei Teile. Der erste Teil vermittelt dem Leser eine Einführung in die Problematik der Rechnernetzwerke allgemein. Darauf aufbauend widmet sich der Autor dem Schwerpunkt seiner Ausarbeitung, dem PC-LAN. Der dritte Teil untersucht das Umfeld von lokalen Rechnernetzen. Die dargelegt Problematik dokumentiert eine große Sachkenntnis des Verfassers auf dem Gebiet der Rechnernetztheroie. Insbesondere die detailierten Ausführungen im zweiten Teil können als außerordentlich praxisorientiert eingeschätzt werden. Nach einer kurzen Beweisführung zur Notwendigkeit des Aufbaus von PC-LAN-Systemen werden Topologie-Konzepte dargestellt und analysiert. Der Autor konzentriert sich vorwiegend auf die zur Zeit marktbestimmenden Konzepte des Token Ring und Bus-Systeme (Ethernet und dessen Derivate). Ergänzt wird dies durch Aussagen zur IBM-LAN-Strategie. Auch alternative Konzepte des PC-Verbundes in einem LAN werden mit ihren Vor- und Nachteilen diskutiert. Ausgehend vom Quasi-Standardbetriebssystem für PC (MS-DOS, Version 3.1) werden LAN-Betriebssysteme vorgestellt. Einen breiten Raum nimmt der Testbericht von zehn lokalen Netzwerken ein. Das Buch ist für Spezialisten und Einsteiger in die PC-LAN-Problematik empfehlenswert. Bei den zahlreichen Abkürzungen fehlt ein Abkürzungsverzeichnis.

Dr. Th. Weller, Dr. K. Arnold

Amiga-Basic im Detail

Prof. David A. Lien, Amiga Programmierpraxis mit MS-Basic, 394 Seiten, kart., TeWi-Verlag München, 59 DM.
ISBN 3-921803-69-1

Was macht ein frischgebackener Amiga-Besitzer, wenn er von Basic noch keine Ahnung hat? Das Amiga-Basic-Handbuch beschreibt zwar alle Befehle des Amiga-Basic übersichtlich und umfassend – aber ausreichend nur für jemanden, der zumindest schon einmal mit einem Computer und Basic gearbeitet hat. Professor David A. Lien, der sich durch eine ganze Reihe von Büchern einen guten Ruf erworben hat, will in seinem Amiga-Basic-Buch genau jenen helfen, die mit dem Amiga ihren Einstieg in die Computerei wagen. Mit über 60 kleinen Übungen führt er den Anfänger systematisch in die Programmierung des Amiga mit MS-Basic, das dem GW-Basic vom IBM-PC recht ähnlich ist, ein. Es beginnt mit dem Starten des Basic-Interpreters, der Programmeingabe und dem Starten des ersten Programms. In recht lokkerem Stil werden dann alle wichtigen Befehle des Amiga-Basic behandelt, wobei jedes Kapitel einen ganz bestimmten Themenbereich behandelt. Übungsaufgaben helfen den Stoff zu vertiefen, die Lösungen sind im Anhang aufgelistet. Ein umfassendes Stichwortverzeichnis hilft dem „Querleser". Ein für Anfänger auf jeden Fall nützliches Buch für den Einstieg in das MS-Basic des Amiga. *JPL*

Dieter Strauß

Lokale Computernetzwerke

Moderne Kommunikationstechnologie verständlich erklärt

Wie bei allen Technologien gibt es auch im Bereich der lokalen Computernetzwerke eine Vielzahl neuer Verfahren und Begriffe. mc beschreibt die Netzwerktechnik und bahnt für Sie einen Weg durch den Begriffsdschungel.

Nachdem PCs in vielen Bereichen eingesetzt werden, wollen die Anwender Dateien auf andere PCs oder sogar Großrechner übertragen, auf gemeinsame Dateien zugreifen und Drucker von mehreren PCs aus benutzen. Diese Anwenderwünsche können mit einem lokalen Computernetzwerk befriedigt werden.

Ein lokales Computernetzwerk (LAN) ist nichts anderes als ein Transportmechanismus für beliebige Datenströme. In der Funktion entspricht es dem Telefonnetz, das ebenfalls einen Transportmechanismus, jedoch für akustische Signale, bereitstellt. In lokalen Netzwerken ist wie im Telefonnetz mit der bloßen Herstellung einer Verbindung noch nicht gewährleistetet, daß sich die Teilnehmer verstehen. Im Telefonnetz könnten beide Teilnehmer eine andere Sprache sprechen und sich nicht verstehen. In einem LAN könnten sich Computer, die eine unterschiedliche Zeichen-Codierung wie ASCII und EBCDIC benutzen, ohne zusätzliche Software ebenfalls nicht verstehen. Es müssen neben den Bedingungen der Hardwareseite, wie elektrische Pegel, Steckverbinder und Übertragungsrate von der Softwareseite „Sprachregelungen" eingehalten werden, damit eine erfolgreiche Kommunikation stattfinden kann.

Lokale Netzwerke sind in ihrer Ausdehnung auf das Grundstück eines Unternehmens beschränkt. Computernetzwerke, die wesentlich größere Flächen abdecken, werden Weitverkehrsnetze genannt. Mit intelligenten Umsetzern, den Gateways, können lokale Netzwerke mit Weitverkehrsnetzen (WAN) gekoppelt werden. Gateways führen die dazu notwendigen Geschwindigkeits- und Protokollumsetzungen durch. Lokale Netzwerke können untereinander über Brükken gekoppelt werden. Im Gegensatz zu Gateways führen Brücken nur Geschwindigkeitsumsetzungen durch.
Ein Netzwerk besteht aus den Arbeitsstationen (Workstations), dem Netzwerk-Server, dem Übertragungsmedium und den Netzwerkschnittstellen.
Auf den Arbeitsstationen eines Netzes laufen die Anwenderprogramme wie in einem nicht vernetzten Computer ab. Programme und Daten können jedoch überall im Netz abgespeichert werden. Als Arbeitsstationen werden meistens PCs verwendet.
Der Netzwerk-Server, das Herz eines Netzwerks, verschafft den Zugang zu gemeinsam nutzbaren Massenspeichermedien, übernimmt den Transport von Dateien über das Netz, ermöglicht mehreren Teilnehmern die Benutzung eines gemeinsamen Druckers und verschafft Zugang zu externen Kommunikationsdiensten. Um den wachsenden Geschwindigkeitsanforderungen der Benutzer zu entsprechen, muß der Netzwerkserver über einen schnellen Prozessor und eine schnelle Festplatte verfügen. Unter einem AT läuft hier nichts.
Die Leistungsfähigkeit heutiger PC-Netzwerke wird weniger durch die theoretisch möglichen Übertragungsraten und den Netzwerkschnittstellen begrenzt als vielmehr durch langsame Plattenspeicherzugriffe.
Die Netzwerkschnittstelle, oft mit NIU (NIU = Network Interface Unit) bezeichnet, verbindet die einzelnen Stationen mit dem Netzwerk. Diese kann sich auf einer PC-Einsteckkarte oder in einem externen Gerät befinden. An Netzwerkschnittstellen, die in einem eigenen Gehäuse untergebracht sind, können meistens mehrere PCs mit ihrer V.24-Schnittstelle angeschlossen werden. Zur Entlastung des PC-Prozessors sind Netzwerkschnittstellen oft mit eigenen Prozessoren ausgestattet.
Lokale Netzwerke lassen sich nach den Merkmalen Netzwerktopologie, Übertragungsmedium, Übertragungstechniken und Zugangsverfahren (Leitungsprotokoll) charakterisieren.

Netzwerktopologien

Die Netzwerktopologie beschreibt die Struktur des Netzwerkes. Es wird unterschieden zwischen Stern-, Ring-, Bus- und Baumstrukturen. In Netzen mit Sternstruktur befindet sich der Netzwerk-Server im Sternknotenpunkt. Zu allen Arbeitsstationen wird vom Sternknoten eine Stichleitung hingeführt. Es ist der Vorteil dieser Struktur, daß fehlerhafte Verbindungen oder Stationen leicht lokalisiert werden können. Nachteilig ist der hohe Verdrahtungsaufwand. Fällt der Sternknoten aus, so steht das gesamte Netzwerk still.
In Netzen mit Busstruktur werden die Arbeitsstationen über einen Kabeladapter, oft mit TAP (Terminal-Anschlußpunkt) bezeichnet, mit dem gemeinsamen Buskabel verbunden. Von Vorteil sind bei dieser Topologie die kurzen Leitungsführungen. Ein Nachteil dieser Topologie ist, daß bei Unterbrechung des Buskabels fast immer das Netzwerk ausfällt. Es ist schwierig fehlerhafte Stellen zu lokalisieren.
Netze mit Baumstruktur lassen sich auf die Busstruktur zurückführen.
In Netzwerken mit Ringtopologie werden Nachrichten von Station zu Station im Ring „weitergereicht". Ringnetze bestehen aus einer Reihe von einzelnen Punkt-zu-Punkt-Verbindungen. Die Signale der übertragenen Daten laufen nur in die fest vorgegebene Richtung, so daß es einfach ist, diese Signale zu verstärken. Zwischen den einzelnen Stationen können auch verschiedene Übertra-

gungsmedien, wie Lichtwellenleiter und Kabel mit verdrillten Doppeladern, verwendet werden.

Übertragungsmedien

In lokalen Netzwerken werden als Übertragungsmedium, je nach Geschwindigkeits- und Zuverlässigkeitsanforderungen, Kabel mit verdrillten Doppeladern, Koaxialkabel oder Lichtwellenleiter eingesetzt. Verdrillte Kupferdoppeladern sind als erdsymmetrisches Medium gegen elektromagnetische Einstreuungen weitgehend unempfindlich. Durch zusätzliche Abschirmung läßt sich der Schutz gegen Einstreuungen noch erhöhen. Gegenüber Koaxialkabeln, die ein assymetrisches Übertragungsmedium sind, bieten Kabel mit verdrillte Doppeladern den Vorteil, daß bei ihnen wenig Energie von der Sende- zur Empfangsseite (Nahnebensprechen) gelangt. Beste Störsicherheit bieten Kabel aus Lichtwellenleitern. Sie sind allerdings am teuersten und ihre Anschlußtechnik ist nicht flexibel. Beim Installieren des Netzwerks sollte der Kabeltyp verwendet werden, der vom Lieferant empfohlen wird. Sowohl Kabel mit verdrillten Doppeladern als auch Koaxialkabel können beträchtliche Unterschiede in ihren elektrischen Eigenschaften besitzen. Die Werbeaussage „einfache Verkabelung mit Telefonleitungen" bezieht sich oft auf Telefonleitungen nach USA-Standard.

Übertragungstechniken

In der Welt der lokalen Computernetzwerke wird zwischen den zwei Übertragungstechniken Basisbandverfahren und Breitbandverfahren unterschieden. Ein Basisbandnetz stellt auf einer Leitung nur einen Übertragungskanal zur Verfügung, so daß den angeschlossenen Stationen das Übertragungsmedium immer nur für eine kurze Zeit zur Verfügung gestellt werden kann. Diese Übertragungstechnik wird als digitales Zeitmultiplex-Verfahren bezeichnet. Basisbandnetzwerke lassen sich preisgünstig realisieren, da keine Modems gebraucht werden. Basisbandnetzwerke sind z.B. Arcnet (Standard Microsystems), Ethernet (DEC, Intel, Xerox), IBM Token-Ring Netzwerk, SK-Net (Schneider und Koch) und Starlan (AT&T).
In Breitbandnetzwerken wird die Bandbreite des Übertragungsmediums in mehrere Kanäle unterteilt. Durch das Frequenzmultiplex-Verfahren können Kanäle mit unterschiedlichen Übertragungskapazitäten und Diensten zur Verfügung gestellt werden. Es ist möglich Sprache, Video und Daten gleichzeitig zu übertragen. An jedem Anschlußpunkt muß ein Modem verwendet werden, das auf den entsprechenden Kanal abgestimmt ist. Breitbandnetzwerke sind z.B. das Wangnet und das IBM PC-Netzwerk.

LAN-Zugangsverfahren

Zwei Zugangsverfahren, auch Leitungsprotokolle genannt, sind weit verbreitet: CSMA und Token-Passing.
Ein Vorläufer heutiger CSMA-Netzwerke wurde bereits in den sechziger Jahren auf Hawaii entwickelt. Die dortige Universität wollte ihren Instituten, die auf mehreren Inseln verstreut sind, Zugang zu einem Großrechner verschaffen. Die Terminals der Institute sollten über eine Funkverbindung mit dem Großrechner kommunizieren. Jede Station durfte senden sobald der „Äther" frei war. Im Aloha-Netz wurde zum ersten Mal das CSMA-Protokoll angewendet.
CSMA (Carrier Sense Multiple Access) ist kurz ausgedrückt ein höre-zu-bevor-du-sprichst-Protokoll. Ein Netz-Teilnehmer darf erst senden, wenn auf dem Übertragungsmedium kein Datenverkehr stattfindet. Da sich die Daten in dem Übertragungsmedium mit endlicher Geschwindigkeit bewegen, kommen die Signale bei den Stationen verzögert an. Unter ungünstigen Bedingungen merken entfernte Stationen nicht, daß die Leitung bereits belegt ist und fangen an zu senden. Es kommt es zur Kollision von Datenpaketen. Netzwerke nach dem CSMA-Protokoll werden daher zusätzlich mit einem Mechanismus ausgerüstet, der Datenkollisionen erkennt und die beteiligten Stationen abschaltet. Das Leitungsprotokoll wird CSMA/CD genannt, wobei CD für Kollissons-Erkennung (Collision Detection) steht. Nach einer Datenkollission senden die Stationen, welche die Kollision erkannt haben, noch einen Moment weiter, damit auch die anderen Stationen die Kollission erkennen können. In der Fachsprache wird dieser Vorgang „Jamming" genannt. Der Begriff „Jamming" ist ein gutes Beispiel, daß im anglo-amerikanischen Sprachbereich technisch anspruchvolle Vorgänge nicht in gestelzten

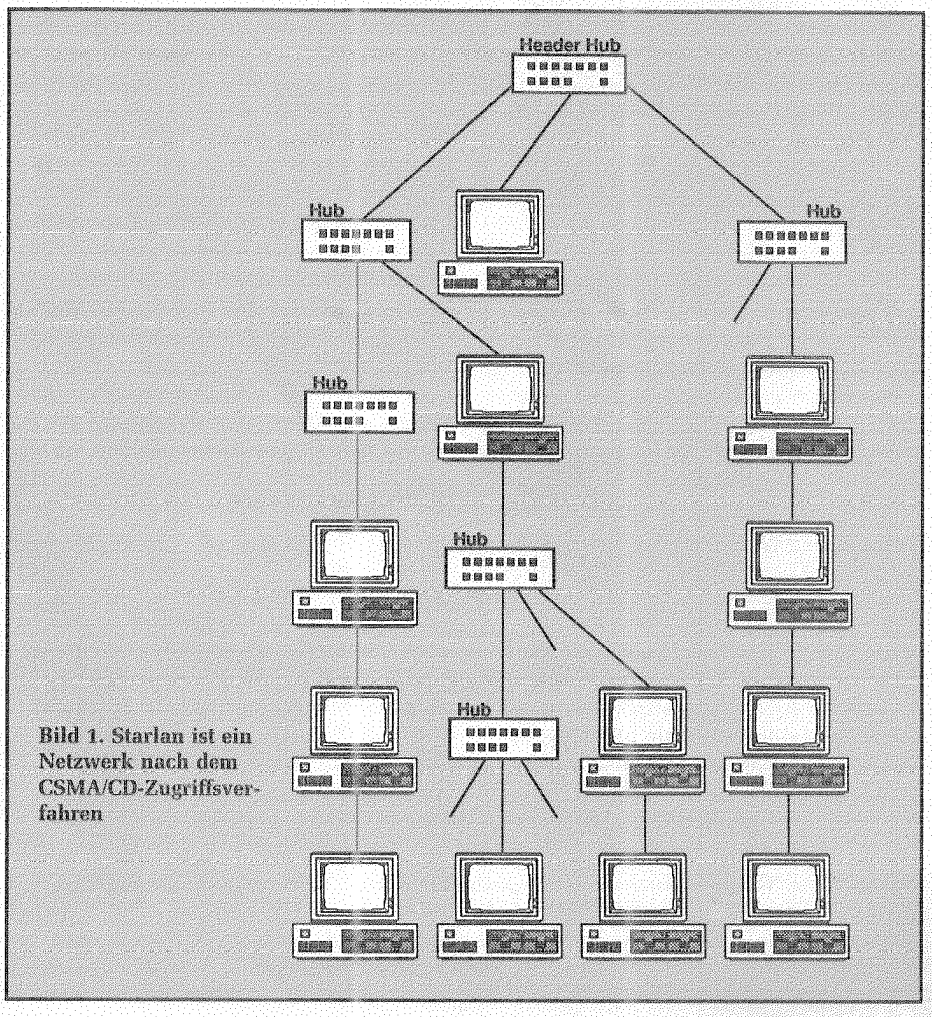


Bild 1. Starlan ist ein Netzwerk nach dem CSMA/CD-Zugriffsverfahren

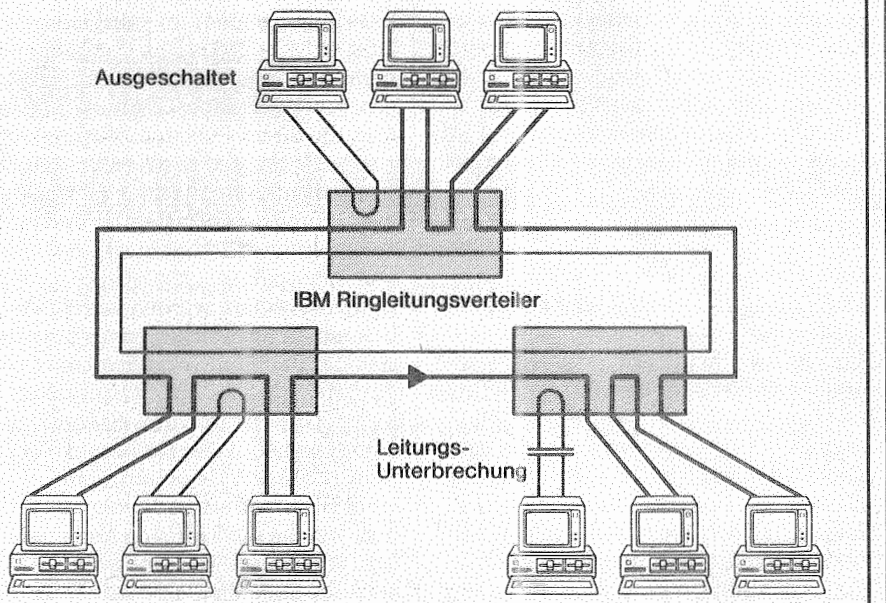


Bild 2. Das IBM Token-Ring-Netzwerk wird sternförmig verkabelt

Worten beschrieben, sondern durchaus humorvoll benannt werden (jam = Marmelade, Gedränge, Störung). In jeder der betroffenen Stationen läuft, nachdem die Kollission erkannt wurde, ein mit einer Zufallszahl initialisierter Zähler ab. Sobald der Zähler abgelaufen ist, versucht die Station erneut ihre Nachricht abzusenden.
Die Wahrscheinlichkeit, daß Datenkollissionen auftreten, ist um so größer, je

Start-Begrenzer 1 Byte	Zugriffs-kontrolle 1 Byte	End-Begrenzer 1 Byte

Bild 3. Das Frei-Token besteht aus drei Byte

mehr Stationen aktiv im Netz arbeiten. Bei Netzwerken mit dem CSMA-Protokoll kann die Zeit, die zum Übermitteln einer Nachricht gebraucht wird, nur statistisch angegeben werden. Das CSMA-Protokoll wird daher in der Fachsprache als nicht deterministisch bezeichnet. Netzwerke mit dem CSMA-Protokoll eignen sich nicht für Echtzeitprozesse.

In Büroumgebung sind sie uneingeschränkt verwendbar. Das bekannteste CSMA/CD-Netzwerk ist Ethernet (Intel, Xerox, DEC). Weitere CSMA/CD-Netze sind z.B. Net/one (Ungermann Bass), Wangnet (Wang), Starlan (AT&T, Western Digital) und das SK-Net (Schneider und Koch). *Bild 1* zeigt Starlan in der Konfiguration mit externen Netzknoten.

Token-Passing

Netze mit dem Token-Passing-Verfahren können in der Bus- oder der Ringstruktur verwirklicht werden. Je nach Struktur werden die Netze Token-Bus oder Token-Ring bezeichnet. Wir beschreiben hier das Token-Passing-Verfahren am Beispiel des IBM-Token-Ring-Netzwerks (*Bild 2*). In Token-Passing-Netzwerken wird ein bestimmtes Bitmuster, das Token, von einer Station zur anderen weitergereicht. Das Token ist ein Datenpuffer mit Adreß- und Steuerzeichen. Mit dem Steuerzeichen wird signallisiert, ob das Token frei oder belegt ist. Sendeberechtigt ist nur die Station, die gerade ein Frei-Token (*Bild 3*) erhalten hat. Empfängt eine Station, die eine Nachricht senden will, das Frei-Token, so wird dessen Bitmuster in das Belegt-Token-Format (*Bild 4*) umgeändert. Die Nachricht wird an dieses aktive Token angehängt. Innerhalb der Nachricht steht im Adressenfeld die Adresse der Empfängerstation. Jede Station die das Belegt-Token erhält, schaut im Adressenfeld nach, ob diese Nachricht für sie bestimmt ist. Ist diese Nachricht nicht für sie bestimmt, so gibt sie einfach das Belegt-Token und die angehängte Nachricht an die nächste Station weiter. Kommt die Nachricht bei der Empfängerstation an, so kopiert diese die Nachricht in ihren Speicher, kennzeichnet die Nachricht als erhalten und gibt diese weiter. Die Nachricht kommt daher nach einem Umlauf wieder bei der Sendestation an. Diese wandelt das Belegt-Token in ein Frei-Token um, wenn die Empfängerstation ihr Nachricht-erhalten-Zeichen gesetzt hat.
In Token-Ring-Netzwerken kann eine endliche Wartezeit garantiert werden, bis eine Station ihre Nachricht absenden kann. Sie sind deterministisch und können auch in Echtzeitprozessen eingesetzt werden.
Das IBM-Token-Ring-Netzwerk entspricht den Standards ECMA 89 und IEEE 802.2/802.5. Es ist ein Basisbandnetz mit einer Übertragungsgeschwindigkeit von 4 MBit/s. Bis zu 260 Stationen können angeschlossen werden. Eine interessante Topologie wurde verwendet – physikalisch gesehen hat es eine Sternstruktur – logisch gesehen eine Ringstruktur. Jedes Kabel zu einem Endgerät (z.B. ein PC) hat je einen Pfad für die Hin- und die Rückrichtung. Im Sternpunktpunkt des Netzes sitzt jedoch ein Ringleitungsverteiler! Durch die physikalische Sternstruktur können Fehler in den Zweigen schnell lokalisiert werden. Damit bis zur Reparatur nicht das ganze Netzwerk lahmgelegt ist, kann ein fehlerhafter Zweig am Sternpunkt einfach überbrückt werden.

Der Chipsatz von Texas Instruments

Um das IBM-Token-Ring-Netzwerk auch für andere Hersteller offen zu halten, hat IBM zusammen mit Texas Instruments einen Chipsatz entwickelt, der von beiden Firmen hergestellt wird. Texas Instruments liefet diesen Chipsatz auch an interessierte Hersteller von LAN-Adapter-Karten. IBM verfolgt damit eine Politik, die bei diesem Netzwerk sicher zu einer breiter Palette an Anbietern führen wird. Der Chipsatz besteht aus fünf Bausteinen: dem Schnittstellenbaustein zum Systembus, kurz SIF (SIF=System Interface) genannt (TMS 38030), dem Kommunikationsprozessor (TMS

Start-Begrenzer 1 Byte	Zugriffs-kontrolle 1 Byte	Rahmen-Kontrolle 1 Byte	Empfänger-Adresse 6 Byte	Absender-Adresse 6 Byte	Datenfeld	Prüfsumme 4 Byte	End-Begrenzer 1 Byte	Status des Rahmens 1 Byte

Bild 4. Aufbau des Belegt-Tokens

38010), dem Protokoll-Handler (TMS 38020) und zwei Bausteinen zur Ringschnittstelle, dem Transceiver TMS38051 und dem Controller TMS38052. Der Baustein TMS 38030 verbindet den LAN-Adapter-Bus mit dem Systembus. Mit zwei Pins kann die Busschnittstelle für die Prozessoren der 680XX-, der 808X-Familie und die Busbreite (8/16 Bit) konfiguriert werden. Daten können über den integrierten DMA-Kanal zwischen dem SIF und dem Speicher des PCs ausgetauscht werden. Der Kommunikationsprozessor ist eine dedizierte 16-Bit-CPU mit integriertem RAM-Puffer. Der Protokoll-Handler ist das Bindeglied zwischen den beiden Ringschnittstellenbausteinen und dem LAN-Adapter-Bus. Er führt die Aufgaben durch, die ein Mikroprozessor nicht schnell genug durchführen kann. Dazu gehören z.B Codierung und Decodierung nach dem Manchesterverfahren und CRC-Berechnungen und -Kontrolle.

LAN-Standards

Um einen Rahmen für die Normung von Kommunikationsprotokollen und von lokalen Netzwerken zu schaffen, wurde von ISO, einer internationalen Normungs-Organisation, die Architektur für offene Systeme (OSI) spezifiziert. Dieser Rahmen wird als ISO-Referenzmodell für offene Systeme bezeichnet. Die sieben Schichten des Referenzmodells heißen:
Bitübertragungsschicht (Physical Layer)
Sicherungsschicht (Data Link Layer)
Vermittlungsschicht (Network Layer)
Transportschicht (Transport Layer)
Kommunikationssteuerungsschicht (Session Layer)
Darstellungsschicht (Presentation Layer)
Anwendungsschicht (Application Layer)

Die Bitübertragungsschicht und ein Teil der Sicherungsschicht sind der Hardware, die restlichen Schichten der Software zugeordnet. *Bild 5* stellt das Schichtenmodell schematisch dar. Transitsysteme sind Übertragungungs- und Vermittlungsvorrichtungen.

Das ISO-Referenzmodell für offene Systeme

Die Bitübertragungsschicht ist die Schicht des physikalischen Übertragungsmediums. Sie beschreibt die elektrischen Pegel der Signale und die mechanischen Verbindungen zwischen den beteiligten Systemen. In der Norm zur V.24-Schnittstelle beschreibt die Bitübertragungschicht die Form und Belegung der 25poligen D-Sub-Steckverbinder und die elektrischen Pegel für logische Nullen und Einsen.
Während jeder Datenübertragung können durch Übersprechen, Leitungsreflexionen und Einstreuungen Daten verfälscht am Empfänger ankommen. Die Sicherungsschicht erkennt und behebt Übertragungsfehler. Weitere wichtige Aufgaben sind die Flußregelung (flow control) und in lokalen Netzen die Abwicklung des Zugangsprotokolls zum physikalischen Medium. Die Flußregelung paßt die Sendegeschwindigkeit an die Empfangsgeschwindigkeit an.
Zur Fehlerbehandlung und Flußkontrolle werden die zu übertragenden Daten in Blöcke (frames) aufgeteilt. Mit Prüfsummenverfahren werden Übertragungsfehler erkannt. In lokalen Netzen wird die Sicherungsschicht in die beiden Teilschichten LLC (Logical Link Controll) und MAC (Media Access Controll) unterteilt. Fehlererkennung und Flußregelung werden in LLC, das Zugangsprotokoll in MAC abgewickelt. Das Zugangsprotokoll ist zuständig für die Vermeidung oder Auflösung von Kollisionen, die beim gleichzeitigen Sendewunsch mehrerer Stationen auftreten können.

Die Vermittlungsschicht wählt den Weg aus, den die Datenpakete zwischen den Endssystemen (Absender und Empfänger) im Netzwerk nehmen müssen. Die beiden unteren Schichten legen nur Protokolle zwischen zwei unmittelbar benachbarten Systemen fest. Zwei Dienste können dem Benutzer von dieser Schicht angeboten werden: Datagramm und virtuelle Verbindung (virtual circuit). Im Datagrammdienst wird jedes Datenpaket isoliert von anderen Datenpaketen durch das Netzwerk zum Empfänger geschickt. Zwischen dem Absender und dem Empfänger wird keine Verbindung aufgebaut. Datenpakete können auf unterschiedlichen Wegen zum Empfänger gelangen. So kann ein Datenpaket einen kürzeren Weg durch das Netzwerk nehmen als die vorausgegangenen und früher als diese beim Empfänger eintreffen. Nacheinander abgeschickte Datagramme können daher beim Empfänger in einer anderen Reihenfolge eintreffen. Es wird nicht garantiert, daß das Datenpaket beim Empfänger ankommt. Die Transportschicht muß die Datagramme in der richtigen Reihenfolge sortieren.

Bei der virtuellen Verbindung wird, bevor ein Datenaustausch stattfinden kann, eine Verbindung aufgebaut. Alle Datenpakete folgen dem Weg, der während des Verbindungsaufbaus vom Netzwerk ausgewählt wurde. Die Datenpakete treffen damit in der richtigen Reihenfolge beim Empfänger ein. Da die Datenpakete bei der virtuellen Verbindung numeriert sind, bemerkt der Empfänger den Verlust eines Pakets und

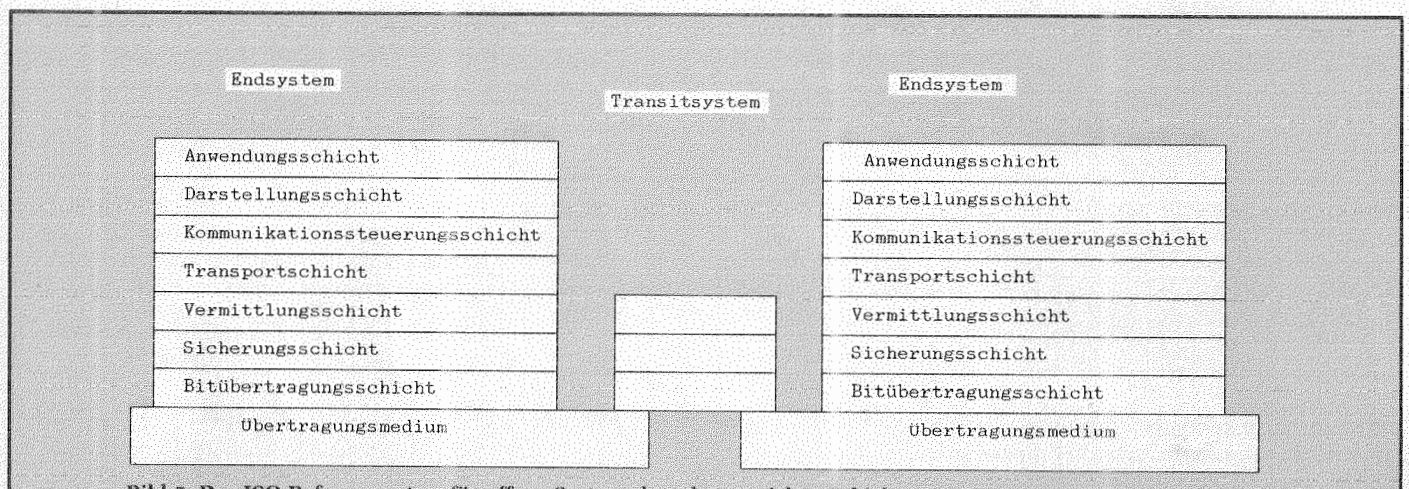


Bild 5. Das ISO-Referenzsystem für offene Systeme besteht aus sieben Schichten

kann den Sender veranlassen dieses Paket noch einmal zu senden.
Datagramme werden meistens in lokalen Netzwerken angewendet, da hier nur wenig Übertragungsfehler auftreten, während virtuelle Verbindungen in Weitverkehrsnetzen angeboten werden. Die Transportschicht liefert zuverlässige Verbindungen zwischen Prozessen in den Endsystemen und stellt sicher, daß die empfangenen Daten richtig sind. Der Transportschicht bleibt die Art des darunterliegendes Netzes verborgen. Sie ist mit der von Multitasking-Betriebssystemen bereitgestellten Schnittstelle zur Interprozeßkommunikation vergleichbar.

Die oberen drei Schichten

Die unteren vier Schichten des ISO-Referenz-Modells sind aus der systematischen Verbesserung und Erweiterung der Eigenschaften des Transportmediums entstanden. Die oberen drei Schichten orientieren sich daran wie Anwendungsprozesse auf Dienste des Betriebsystems zugreifen. Die drei oberen Schichten können so umschrieben werden, daß für jede geregelte Kommunikation gewisse formale Umgangsformen, ein Sprechen der gleichen Sprache und ein Denken in gleichen Begriffen erforderlich sind.

Die Kommunikationssteuerungsschicht stellt Sprachmittel zur Verfügung, um eine Kommunikationsbeziehung (=Sitzung) zu eröffnen, durchzuführen, zu steuern und wieder geordnet zu beenden. Diese Schicht stellt Schnittstellen zwischen dem Betriebssystem, der Anwendung und der Transportschicht bereit. Prozeßnamen werden Netzwerksadressen zugeordnet und virtuelle Verbindungen werden auf- und abgebaut.

Bei der Kommunikation ist außer geregelten Umgangsformen erforderlich, daß beide „Partner" sich verstehen, also die gleiche Sprache sprechen. Die Darstellungsschicht führt Codeumwandlungen, Umformatierungen, Datenkompression und -expansion und Syntaxumwandlungen durch.

In der Anwendungsschicht sind die Teile eines Anwendungsprozesses enthalten, die sich mit der Rechner-Kommunikation befassen. Die Anwendungsschicht stellt für den Anwendungsprozeß ein Fenster zu dem Partner-Prozeß dar, mit dem eine Kommunikationsbeziehung besteht.

Glossar

Basisbandübertragungssystem: Die untere Grenzfrequenz ist theoretisch 0 Hz. Während eines Zeitpunkts kann nur eine Nachricht übertragen werden

Breitbandübertragungssystem: Die Signale werden moduliert. Auf dem Übertragungsmedium können mehrere Nachrichten gleichzeitig übertragen werden

Brücke (Bridge): Verbindung zwischen Netzwerken. Eine Brücke führt Geschwindigkeitstransformationen durch

BSC: Binary Synchronous Communication. Synchrones Übertragungsprotokoll von IBM. Wird häufig Bisync genannt

Bustopologie: Linienstruktur. Alle Netzwerkteilnehmer hören alle Nachrichten. Eine Nachricht wird direkt von der Nachrichten-Quelle zum Ziel gesandt

CCITT: Comité Consultatif International Télégraphique et Téléphonique

CSMA/CA: Carrier Sense Multiple Access with Collision Avoidance

CSMA/CD: Carrier Sense Multiple Access with Collision Detection

DCE: Data Circuit Terminating Equipment. Datenübertragungseinrichtung (DÜE), z.B. ein Modem

DTE: Data Terminal Equipment. Datenendeinrichtung (DEE), z.B. ein PC

ECMA: European Computer Manufacturers' Association

Gateway: Intelligenter Schnittstellenverbinder der Aufgaben der Geschwindigkeits- und Protokolltransformation übernimmt. Gateways werden zur Verbindung von lokalen Netzwerken mit Weitverkehrnetzen eingesetzt

HDLC: High Level Data Link Control. Von CCITT spezifiziertes bitsynchrones Datenübertragungsprotokoll

Hub: Netzwerkknoten

IEEE: Institute of Electrical and Electronic Engineers. Amerikanischer Fachverband

ISDN: Integrated Services Digital Network

ISO: International Organisation for Standardization. Internationales Normungsgremium

Kollision: Beim gleichzeitigen Übertragen von mehreren Datenpaketen auf dem gleichen Übertragungskanal werden die Daten verstümmelt

LAN: Local Area Network. Lokale Computer-Netzwerke sind in ihrer Ausdehnung auf das Privatgelände eines Unternehmens beschränkt

LLC: Logical Link Controll

MAC: Media Access Control

Manchester Codierung: Codierungsart bei der aus den Signalflanken des Datenstroms der Übertragungstakt zurückgewonnen werden kann

MAP: Manufacturing Automation Protocol. Von General Motors initiiertes Netzwerk im Fertigungsbereich

Netbios: BIOS von IBM das Netzwerkkommandos unterstützt

NetWare: Netzwerk von Novell. Novells Netzwerksoftware wird von vielen Herstellern adaptiert

OSI: Open Systems Interconnection. Um Computer unterschiedlicher Hersteller miteinander kommunizieren zu lassen, wurde von ISO der OSI-Standard definiert. Systeme, die diesen Standard beherrschen, werden als offene Systeme bezeichnet

PBX: Private Branch Exchange. Telefon-Nebenstellenanlage

Protokoll: Eine Reihe von Regeln, die den Datenaustausch über ein Netzwerk beschreiben

Record locking: Funktion, mit der Zugriffe auf Regionen innerhalb einer Datei gesperrt werden können

Ringtopologie: Nachrichten werden von Station zu Station im Ring „weitergereicht"

SDLC: Synchronous Data Link Control. Von IBM spezifiziertes bitsynchrones Übertragungsprotokoll

SNA: Systems Network Architecture. Von IBM spezifiziert

Sterntopologie: Alle Nachrichten laufen über die zentrale Knotenstation

TAP: Terminal-Anschluß-Punkt

Token: Zeichen oder Zeichenfolge

Topologie: Netzwerkstruktur. Es wird unterschieden zwischen Stern-, Ring-, Bus- und Baumstrukturen

WAN: Wide Area Network. Weitverkehrsnetze sind fast immer öffentliche Netze, die der Posthoheit unterliegen. WANs werden in Zukunft von ISDN geprägt

Workstation: Arbeitsstation. Arbeitsplatzcomputer

Folgende Dienste können zur Verfügung gestellt werden:

File-Server
Druck-Server
Elektronische Post
Übertragung von Dateien (File Transfer)
Virtuelles Terminal
Übertragung und Verwaltung von Aufträgen (remote job entry)

Von den Standardisierungsorganisationen IEEE und ECMA wurden drei Standards für lokale Netze definiert. Die Standards beziehen sich auf die beiden unteren Schichten des ISO-Referenzmodells für offene Systeme, die Bitübertragungsschicht und die Sicherungsschicht.
Es wurden der CSMA/CD-Bus (IEEE 802.3), der Token Bus (IEEE 802.4) und der Token Ring (IEEE 802.5) standardisiert. Der Standard der Sicherungsschicht heißt IEEE 802.2. Er gilt für alle drei Verfahren.

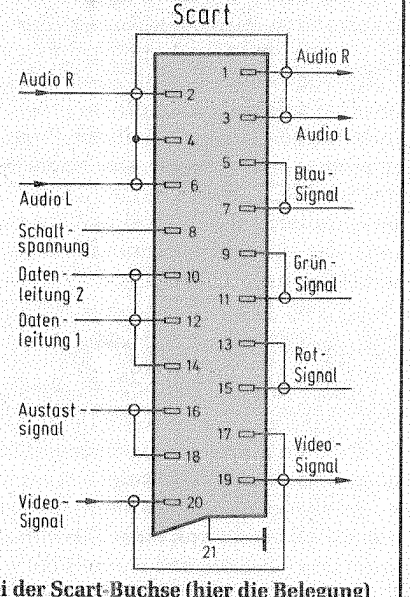
Netzwerkfunktionen unter MS-DOS

Im PC-Bereich werden erst mit MS-DOS 3.1 elementare Funktionen zur Unterstützung von Netzwerken bereitgestellt. In früheren DOS-Versionen können dem Anwender vom File-Server nur logische Laufwerke zur Verfügung gestellt werden. Dürfen mehrere Anwender auf eine Datei simultan (file sharing) zugreifen, so muß es einen Mechanismus geben, der ihre Zugriffe so synchronisiert, daß kein Benutzer die Daten des anderen beschädigen kann. DOS 3.1 stellt hierzu eine Funktion zum Sperren und Freigeben von Dateizugriffen zur Verfügung (record locking).
Mit der DOS-Funktion 5CH läßt sich ein Teil einer Datei (Region) freigeben oder sperren. Mit dem Register Al wird angegeben, ob die Region gesperrt (AL=0) oder freigegeben (AL=1) werden soll. BX enthält die Dateinummer (Handle) und die Register CX und DX geben den Byte-Offset der zu sperrenden oder freizugebenden Region an. Die Register SI und DI enthalten die Länge der Region in Bytes. Mit INT 21H wird die Funktion ausgeführt.
Vernetzte PCs können sowohl auf lokale Resourcen (Dateien, Drucker) als auch auf Netz-Resourcen zugreifen. Jeder netzwerkfähige PC braucht daher ein Programm, das die Zugriffe in die richtige Richtung leitet. Dieses Programm wird Netzwerk-Interface-Schale oder Redirektor genannt.
Bei netzweiten Zugriffen zwingt DOS 3.1 alle Anwendungen auf den File-Server zu gehen. Auf einer elementaren Ebene wird damit die Datenintegrität sichergestellt.
DOS 3.1 bietet keine komfortable Netzwerkfunktionen an. Es schafft jedoch die Voraussetzung komfortablere Systeme auf DOS aufzusetzen. Zwei Netzwerkprodukte sind weit verbreitet: Microsofts MS-NET und Novells Netware. Beide Produkte werden hauptsächlich an OEMs ausgeliefert und von diesen veredelt.
Die Entwicklung der Netzwerktechnik ist noch lange nicht abgeschlossen. mc wird für Sie „am Ball bleiben" und über Neuerungen berichten. Im Glossar finden Sie kurze Erläuterungen zu Abkürzungen und häufig verwendeten Begriffen der Kommunikations- und Netzwerktechnik.

Rechner an der Scart-Buchse

An der Scart- oder Euro-AV-Buchse, die viele der heutigen Farbfernseher besitzen, läßt sich nicht nur der Video-Recorder anschließen – oft kann man damit auch das Fernsehgerät als FBAS- oder RGBA-Monitor für den Hausgebrauch benutzen.
Im *Bild* ist die Belegung der Buchse dargestellt. Da man bei FBAS (d. h. „Farb-Bildinhalt-Austast-Signal") die durch die Bildröhre festgelegte maximale Auflösung kaum erreichen kann, sollte man man den Fernseher – soweit möglich – als RGBA- (d. h. Rot-Grün-Blau-Analog-) Monitor betreiben. Allerdings sind lange nicht alle Fernsehempfänger, die mit einer AV-Buchse ausgestattet sind, auch wirklich auf die Wiedergabe von RGBA-Signalen eingerichtet: Da hilft nur ein Blick ins Schaltbild oder ins Gerät.
Die Umschaltung zwischen RGB und FBAS erfolt meist automatisch: Es wird die Signalquelle gewählt, die ein Signal anbietet. Bei anderen Fersehern muß man zwischen zwei AV-Stellungen wählen.
Für die RGBA-Wiedergabe muß man über Spannungsteiler die Pegel anpassen: Während die TTL-Ausgänge eines Computers Low- und High-Pegel (also 0...4 V) liefern, soll die maximale Eingangsspannung am Rot-, Grün-, Blau- und Austastsignal-Eingang jeweils ein Volt betragen. Liegen am Computer die horizontalen und vertikalen Synchronisations-Impulse getrennt vor, so kann man sie bei dieser Gelegenheit gleich zusammenmischen. Wichtig für die Dimensionierung der Spannungsteiler: Der Eingangswiederstand dieser Anschlüsse beträgt 75 Ω.
Leider werden die Belegungen der Anschlüsse 8, 10 und 12 von den Herstellern recht verschieden interpretiert: Legt man z. B. eine Schaltspannung von 12 V an Anschluß 8 eines ITT-Digivision, so schaltet der Ferseher vom laufenden Programm automatisch auf AV (also das Computerbild), fällt die Spannung auf 0 V, schaltet er zurück. Grundig-Fernseher nehmen die 12 V zur Kenntnis (und zeigen eventuell das Vorhandensein an), ziehen aber keinerlei Konsequenz daraus. Bis auf die genormten Spannungen (0 oder 12 V) läßt sich allgemein nur sagen, daß der Betrieb auch ohne Schaltspannung möglich ist.
Ähnlich herstellerspezifisch sind die Datenleitungen an den Anschlüssen 10 (Clock) und 12 (Data): Sie führen TTL-Pegel (0...5 V) und liefern vielleicht Anweisungen der Fernbedienung für den Video-Recorder (in welchem Format auch immer), vielleicht aber auch Nichts.
Der Vollständigkeit halber noch die übrigen Pegel: Die Audio-Ausgänge liefern eine Effektivspannung von 100 mV bei einem Innenwiderstand von 1 kΩ, die Eingänge erwarten den gleichen Pegel bei einem Eingangswiderstand von 10 kΩ. FBAS-Ein- und Ausgang sind intern mit 75 Ω abgeschlossen; 0 V entspricht hier Synchronisations-Pegel, 0,3 V entspricht schwarz und 1 V weiß.

Ulrich Kruppe



Bei der Scart-Buchse (hier die Belegung) ist es wie bei der V.24-Schnittstelle: Man weis nie, welche Anschlüsse im Einzelfall wirklich bedient werden...

Dr. Helmut Merz

Micro-Prolog

Künstliche Intelligenz für Mikrocomputer

Die Programmiersprache Prolog, zum ersten Mal 1973 in Marseille implementiert, kann heute in Europa und Japan als die wichtigste Sprache der „künstlichen Intelligenz" angesehen werden. Mit Micro-Prolog läßt sich schon seit einiger Zeit auch Mikrocomputern ein gewisses Maß an Intelligenz vermitteln.

Ein intelligentes System ist unter anderem dadurch charakterisiert, daß es in der Lage ist, ohne explizite Anleitung Probleme zu lösen. Computer sind demnach ziemlich dumm: Der Programmierer muß jeden Arbeitsschritt, den die Maschine ausführen soll, genau vorgeben. Allerdings läßt sich diese Borniertheit des Computers leicht abmildern (wenn auch nicht beseitigen), indem man höhere Programmiersprachen verwendet.

Folgt man dem Gedanken, daß der Computer selbst entscheiden soll, wie ein Problem zu lösen ist, so gelangt man zum Prinzip der deskriptiven (d. h. beschreibenden) Programmierung. Ein deskriptives Programm wäre demnach einfach eine Beschreibung des Problems und nicht, wie bei der herkömmlichen imperativen (also befehlsorientierten), prozeduralen Programmiertechnik, eine Folge von Anweisungen. Einen Versuch, zu einer solchen deskriptiven Programmierung zu gelangen, stellt die Programmiersprache Prolog dar. Der Name bedeutet „PROgrammieren in LOGik": Die Beschreibung erfolgt in einem Formalismus, der aus der Prädikatenlogik entwickelt wurde.

Beschreibung eines Fahrrades

Im *Bild 1* ist ausschnittweise eine auf dem Drucker protokollierte Daten-Eingabe mit dem Micro-Prolog-Interpreter gezeigt. Das Prolog-Programm – wenn man es überhaupt so nennen kann; man sollte eigentlich besser von einer Datenbasis sprechen – besteht aus einer Folge von Feststellungen (Assertionen), die mit Hilfe des Kommandos „add" eingegeben werden. *Bild 2* zeigt, was man erhält, wenn man das im folgenden verwendete

```
add (Beleuchtung TEIL-VON Fahrrad)

add (Rahmen TEIL-VON Fahrrad)
                :
                .
```

Bild 1. Aufbau einer Micro-Prolog-Datenbasis. Die Benutzer-Eingaben sind – wie auch in den folgenden Abbildungen – unterstrichen

```
list TEIL-VON
Beleuchtung TEIL-VON Fahrrad
Rahmen TEIL-VON Fahrrad
Antrieb TEIL-VON Fahrrad
Rad TEIL-VON Fahrrad
Dynamo TEIL-VON Beleuchtung
Scheinwerfer TEIL-VON Beleuchtung
Ruecklicht TEIL-VON Beleuchtung
Kabel TEIL-VON Beleuchtung
Birne TEIL-VON Scheinwerfer
Gehaeuse TEIL-VON Scheinwerfer
Birne TEIL-VON Ruecklicht
Gehaeuse TEIL-VON Ruecklicht
Felge TEIL-VON Rad
Speiche TEIL-VON Rad
Reifen TEIL-VON Rad
Schlauch TEIL-VON Reifen
Ventil TEIL-VON Reifen
Mantel TEIL-VON Reifen
```

Bild 2. Mit „list" kann man sich die gesammte Datenbasis ansehen

Beispiel mit dem Kommando „list" auf dem Bildschirm ausgibt.

Die in der Datenbasis enthaltenen Informationen erklären sich selbst: Wir haben es mit einer Aufstellung der Teile zu tun, aus denen sich ein Fahrrad zusammensetzt. Dabei kommt dem Wort TEIL-VON besondere Bedeutung zu: Es verknüpft jeweils zwei Objekte – z. B. Beleuchtung und Fahrrad – und gibt somit eine Relation (Beziehung) zwischen diesen Objekten wieder. Wie könnte man nun die Tatsache formulieren, daß das Rücklicht nicht nur ein Teil der Beleuchtung ist, sondern auch indirekt ein Teil des Fahrrades? Dies wird in *Bild 3* illustriert: Die neue Relation „Teil-von" (die Unterschiede zwischen Groß- und Kleinschreibung werden bei Micro-Prolog beachtet) gibt diese allgemeinere Beziehung wieder. Die beiden Assertionen zu Teil-von unterscheiden sich von denen zu den bisherigen TEIL-VON-Feststellungen: Während es sich bei letzteren um einfache Fakten handelt, haben wir es nun mit Regeln zu tun; außerdem tauchen nun die Buchstaben X, Y und Z statt den Namen von Objekten auf: Die Teil-von-Regeln enthalten Variablen.

Regeln und Variablen

Eine Prolog-Regel hat die Form „eine bestimmte Schlußfolgerung ist wahr (ist als bewiesen anzusehen), wenn diese und jene Bedingungen erfüllt (wahr, bewiesen) sind".

Ersetzt man nun das deutsche „wenn" durch das englische „if" und Schlußfolgerung wie Bedingung durch die hier üblichen Prolog-Sätze, die bereits aus den TEIL-VON-Fakten bekannt sind, so hat man schon die Regel in Micro-Prolog-Schreibweise vor sich. Mehrere Bedingungen werden dabei durch das Wort „and" oder durch das Zeichen „&" verknüpft.

X und Y sind Variablen und keine festen Objekte. Der Unterschied besteht bei Micro-Prolog darin, daß Variablen mit einem der Buchstaben X, Y oder Z (in Groß- oder Kleinschreibung) beginnen, denen eine Zahl folgen kann. Für eine Variable kann man jedes beliebige Objekt einsetzen, etwa Rücklicht oder Fahrrad. Doch stellen wir einfach einmal unsere Frage und sehen, was geschieht.

Fragen an die Datenbasis

Einige der möglichen Fragen (englisch queries) und die Antworten darauf – so wie sie auf dem Schirm erscheinen – sind in *Bild 4* gezeigt. Die erste ist genau

```
X Teil-von Y if
      X TEIL-VON Y
X Teil-von Y if
      Z TEIL-VON Y and
      X Teil-von Z
```

Bild 3. So ist die Relation „Teil-von" definiert

```
is (Ruecklicht Teil-von Fahrrad)
YES

which (x : x Teil-von Reifen)
Schlauch
Ventil
Mantel
No (more) answers

which (x : Ruecklicht Teil-von x)
Beleuchtung
Fahrrad
No (more) answers
```

Bild 4. Mit is- und which-Fragen holt man die gewünschten Informationen aus der Datenbasis

die oben angesprochene Frage „Ist es wahr (beweisbar), daß das Rücklicht ein Teil des Fahrrades ist ?". Die Antwort YES ist zweifellos richtig. Wie aber gelangt der Rechner zu dieser Erkenntnis? Das Verfahren kann man mit dem Begriff „Mustervergleich" (pattern matching) erfassen: Der Interpreter durchsucht die Datenbasis nach einem Muster, das mit der Frage zur Deckung gebracht werden kann. Das erste Muster, das er findet, ist die Schlußfolgerung der ersten Teil-von-Regel, wobei die Variablen X und Y an die Werte „Ruecklicht" und „Fahrrad" gebunden werden. Um eine Regel zu erfüllen, müssen natürlich die entsprechenden Bedingungen überprüft werden. Ist das nicht möglich, so wird nach einer anderen Assertion gesucht, die eine positive Antwort auf die gestellte Frage ermöglicht.

Dieser komplizierte Suchmechanismus braucht den Anwender zum Glück im Allgemeinen nicht zu interessieren, die Hauptsache ist, er beschreibt den Problemraum korrekt. Die weitere Arbeit bleibt dann dem Interpreter überlassen. Dabei können die Fragen noch viel komplizierter sein als die oben gestellte. Zur Syntax von Micro-Prolog ist dabei noch anzumerken, daß bei which-Fragen das, was in der Klammer links vom Doppelpunkt steht, das sogenannte Antwortmuster darstellt. Dieses wird für jede gefundene Antwort ausgegeben, wobei für jedes Vorkommen einer Variablen im Antwortmuster der entsprechende aktuelle Wert aus dem Fragemuster (rechts vom Doppelpunkt) eingesetzt wird.

Micro-Prolog und Standard-Prolog

Die Beispiele sind in Micro-Prolog, einem speziellen Prolog-Dialekt für Mikrocomputer, geschrieben. Nun gibt es zwar keinen formalen Prolog-Standard wie etwa bei Pascal oder Cobol, aber die von Clocksin und Mellish [1] beschriebene Version wird allgemein als ein solcher Standard aufgefaßt. Micro-Prolog bietet nun alle Möglichkeiten der Programmierung, die auch dieser Standard beinhaltet, jedoch mit einer etwas abweichenden Syntax. Die Autoren von Micro-Prolog haben nicht einfach einen Prolog-Interpreter für Mikrocomputer verfaßt, sondern eine eigene Sprachversion geschaffen, die zwei wichtige Eigenschaften besitzt: Zum einen ist Micro-Prolog dahingehend optimiert, seine Programme in möglichst wenig Speicher unterzubringen, zum anderen sollten Micro-Prolog-Ausdrücke besser lesbar sein als ihre Entsprechungen in Standard-Prolog. So findet man in Standard-Prolog statt des aussagekräftigen „if" die etwas seltsame Zeichenkombination ":-", statt des Verbindungswortes „and" steht einfach ein Komma. Ein weiterer Unterschied ergibt sich in der Formulierung von Sätzen: In Standard-Prolog ist man auf die sogenannte Präfix-Form beschränkt, d. h. das Prädikat steht vor einer in Klammern gefaßten Liste von Objekten, im Beispiel also „TEIL-VON (Beleuchtung Fahrrad)" statt „Beleuchtung TEIL-VON Fahrrad".

Letztere Form, bei der das Prädikat zwischen den Objekten steht, bezeichet man als Infix-Form. Man kennt sie aus der Mathematik, wo ja im allgemeinen auch der Operator zwischen den Argumenten zu finden ist. In Micro-Prolog (wie in vielen anderen Prolog-Implementationen) kann man nun zwischen Infix- und Präfix-Form nach Belieben wählen, sofern die Relation zweistellig ist, d. h. wenn sie genau zwei Objekte verknüpft.

Bei einstelligen Relationen (Eigenschaften, z. B. Scheinwerfer defekt) kann man zwischen der Postfix-Schreibweise (Prädikat folgt dem Objekt) und der Präfix-Schreibweise – also „defekt (Scheinwerfer)" – wählen. Relationen mit mehr als zwei Argumenten sind immer in Präfix-Notation anzugeben.

Die Basis-Syntax

Die Möglichkeit, wahlweise Post- bzw. Infix- statt der etwas starren Präfix-Schreibweise zu verwenden, steigert die Lesbarkeit von Mikro-Prolog-Programmen deutlich. Sie wird allerdings mit einem erheblich höheren Speicherplatz-Verbrauch erkauft, denn der Interpreter muß ja die Routinen enthalten, die ihn in die Lage versetzen, die verschiedenen Eingabeversionen korrekt zu verarbeiten. Will oder muß man Speicherplatz sparen, so kann man daher auf diese Möglichkeit des Programmaufbaues und der Formulierung von Fragen verzichten. Dann benötigt der Interpreter unter CP/M-80 nicht mehr als 18 KByte. Die Programmierung muß dann allerdings in der Basis-Syntax erfolgen, einer speziellen Schreibweise mit Ähnlichkeiten zu Lisp: Jede Assertion besteht aus einer Folge von in Klammern gefaßten Listen.

Bild 5 zeigt die beiden Teil-von-Regeln in dieser Basis-Syntax. Im Speicher erfolgt die Darstellung dieser Listen in Form von Zellen, die über Zeiger verknüpft sind (siehe [2]).

```
((Teil-von X Y)
  (TEIL-VON X Y))
((Teil-von X Y)
  (TEIL-VON Z Y)
  (Teil-von X Z))
```

Bild 5. So sehen die beiden „Teil-von"-Regeln in der Basis-Syntax aus

Will man statt dieser Basis-Notation die beschriebene bequemere Prolog-Syntax verwenden, so muß man nach dem Laden des eigentlichen Micro-Prolog-Interpreters noch ein Programm namens Simple hinzuladen, erst dann kann der Dialog in der Weise erfolgen, wie es die Beispiele in Bild 1...4 zeigen. Die Simple-Datei enthält alle Relationen, die zur Übersetzung von Assertionen und Fragen aus der Prolog- in die Basis-Syntax notwendig sind. Das heißt nichts anderes, als daß unsere ganze Fahrrad-Datenbasis von Simple in die Basis-Syntax übersetzt wurde, wo sie platzsparend und zur schnellen Verarbeitung bereitliegt.

Micro-Prolog-Module

Simple besteht aus drei Programm-Modulen. Solche Module erfüllen in Micro-Prolog einen doppelten Zweck: Zum einen sind sie, ganz nach Bedarf, leicht zu laden und leicht wieder zu löschen, man kann also sehr effektiv mit dem kargen Speicherplatz umgehen; zum anderen ermöglichen sie den Aufbau abgeschlossener Programmuntereinheiten mit definierter Schnittstelle nach außen. Das Konzept entspricht weitgehend dem der Programmiersprache Modula-2: Für jedes Modul ist (in der Export-Liste) genau festgelegt, welche Symbole für Programme außerhalb des Moduls zugänglich sind, in der Import-Liste stehen all jene

```
which (x : SUM (23 31 x))
54
No (more) answers

which (x : SUM (45 x 9))
-36
No (more) answers

which (y : SUM (99 147 x) and TIMES (x y 14))
5.6910569E-2
No (more) answers
```

Bild 6. Beispiele für das Arbeiten mit arithmetischen Relationen

Symbole, die sich das Modul von außen holen kann.

Zu Micro-Prolog gehören eine ganze Reihe solcher oft recht nützlicher Module: Von den drei im Programm Simple zusammengefaßten Modulen war bereits die Rede, ansonsten gibt es vor allem Werkzeuge zur automatischen Programmentwicklung, Fehlersuche mittels Tracing und zur automatischen Fehlerbehandlung. Zur Micro-Prolog-Version für IBM-PC und -Kompatible gehört auch ein sogenannter Edinburgh-Supervisor, der das Programmieren in der oben genannten Standard-Prolog-Syntax ermöglicht.

Prolog und Zahlen

Natürlich kann man in Prolog nicht nur mit handfesten Objekten wie Fahrrädern arbeiten, sondern auch mit Zahlen, und zwar sowohl mit Ganzzahlen wie mit Gleitkommazahlen. Dazu passend gibt es einige eingebaute Relationen, mit denen sich arithmetische Aufgaben bewältigen lassen (*Bild 6*).

Hier zeigt sich wieder, daß Prolog keine Sprache ist wie jede andere: Es gibt nämlich keinen Zuweisungsoperator wie in praktisch allen anderen Programmiersprachen (einschließlich Lisp), sondern die Bindung einer Variablen an einen Wert erfolgt nur während einer „query evaluation", während der Suche nach der Antwort auf eine Frage. Ist während eines solchen Suchprozesses eine Bedingung SUM(15 17 X) zu erfüllen, so wird X eben an den Wert 32 gebunden.

Die arithmetischen Relationen in Bild 6 demonstrieren noch eine andere Eigenschaft von Prolog-Relationen im allgemeinen: Sie können auch zur Beantwortung von einer invertierten Fragestellung herangezogen werden. Die Bedingung SUM (X Y Z) ist ja genau dann erfüllt, wenn Z die Summe von X und Y ist. Sind nun die Werte von X und Z bekannt, so muß Y die Differenz von X und Z darstellen, Y wird also an den entsprechenden Wert gebunden. Auf diese Weise lassen sich mit den Funktionen SUM und TIMES auch die Differenz bzw. der Quotient zweier Zahlen errechnen. In ähnlicher Weise wurde im dritten Beispiel in Bild 4 auch ermittelt, von welchen Objekten das Fahrrad-Rücklicht ein Teil ist.

Listen als Objekte

Bild 7 zeigt, wie man die Antworten auf eine Frage in einer Liste sammeln kann, die selbst wieder mit Prolog-Relationen untersucht werden kann. Eine Liste wird einfach als eine Folge von Objekten dargestellt – darunter können natürlich wieder Listen sein –, die in Klammern eingeschlossen ist. In der internen Darstellung im Computer ist eine Liste eine Sammlung von über Zeiger verbundenen Knoten [2].
Um zu erfahren, wieviele Teile das Fahrrad hat, muß nun noch die Länge der betreffenden Antwortliste bestimmt werden. Dazu braucht man eine ganz allgemein sehr nützliche Relation, die in *Bild 8* definiert ist. X Laenge-von Y ist genau dann wahr, wenn der Wert, an den X gebunden ist – das sollte natürlich eine Zahl sein – der Länge der Liste Y entspricht (*Bild 9*).

Um diesen Zusammenhang zwischen einer Liste und ihrer Länge beschreiben zu können, benötigt man genau zwei Assertionen. Die erste ist unmittelbar verständlich: Die Länge einer Liste ohne jedes Element ist natürlich Null. In der zweiten Assertion wird ein neues Hilfsmittel für die Beschreibung von Listen eingeführt: der senkrechte Strich. Was links von diesem Strich steht – das können ein oder mehrere Objekte beliebiger Komplexität sein –, symbolisiert die Elemente, aus denen die Liste am linken Ende (am „Kopf") besteht. Rechts vom Strich – hier muß immer ein einzelnes Element stehen – ist ein Muster für den Rest der Liste, der entsteht, wenn man die links vom Strich stehenden Elemente entfernt.

Die zweite Laenge-von-Relation hat demnach folgenden Sinn: X ist die Länge einer Liste, die aus mindestens einem Element Y und einem Rest Z – der auch die leere Liste sein kann – besteht, wenn gilt, daß es ein x gibt, das die Länge der genannten Restliste Z ist – das ist der Gehalt der ersten Bindung – und dieses x um Eins kleiner ist als X.

Wie die Relation Teil-von ist auch Laenge-von rekursiv, d. h. sie enthält eine Bedingung, zu deren Erfüllung sie selbst wieder aufgerufen werden muß. Dies geht deswegen nicht in alle Ewigkeit weiter, weil jeder Aufruf mit einer um ein Element verkürzten Liste als zweites

```
which (x : x isall (y : y Teil-von Fahrrad))
(Mantel Ventil Schlauch Reifen Speiche Felge Gehaeuse Birne Gehaeuse Birne Kabel
 Ruecklicht Scheinwerfer Dynamo Rad Antrieb Rahmen Beleuchtung)
No (more) answers
```

Bild 7. Mit isall kann man sich eine Liste der gesuchten Teile erstellen lassen

Bild 8. Die rekursive Relation „Laenge-von" liefert die Anzahl der Elemente einer Liste

```
0 Laenge-von ()
X Laenge-von (Y!Z) if
      x Laenge-von Z and
      SUM (x 1 X)
```

```
which (z : x isall (y : y Teil-von Fahrrad) and z Laenge-von x)
18
No (more) answers
```

Bild 9. Der Einsatz der in Bild 8 definierten Relation

Objekt erfolgt, so daß irgendwann einmal die leere Liste überprüft wird. Und deren Länge ist ja bekannt, dafür wurde in der ersten Assertion gesorgt.

Micro-Prolog in der Praxis

Der hier beschriebene Micro-Prolog-Interpreter stammt von Logic Programming Associates Ltd. (LPA) in London und ist u. a. für das Betriebssystem CP/M auf Z80-Rechnern sowie für IBM-PC und -Kompatible unter CP/M-86 und MS-DOS erhältlich. Letztere Version ist gegenüber der Z80-Version rund doppelt so teuer, bietet aber auch einige zusätzliche Möglichkeiten, u. a. den bereits erwähnten Edinburgh-Supervisor für die Arbeit mit der Standard-Prolog-Syntax und ein Hilfsprogramm zum Einbinden von Maschinensprache-Routinen in den Interpreter.

Im Prinzip ist eine solche Erweiterung von Micro-Prolog aber auch bei der Z80-Version möglich, muß dort aber von Hand mittels DDT erfolgen. Die dafür notwendige Symboltabelle, die auch für „Programm-Chirurgen" ganz nützlich ist, wird mitgeliefert. Ebenso wichtig wären meiner Meinung nach allerdings eingebaute Relationen, die unmittelbare Speicher- und Schnittstellen-Zugriffe (wie PEEK und POKE in Basic) und BDOS- bzw. BIOS-Aufrufe wie unter Turbo-Pascal erlauben.

Bezüglich der Bildschirmausgabe ist positiv zu vermerken, daß auch Steuerzeichen ausgegeben werden können und somit eine Formatierung möglich ist. Hinsichtlich der Disketten-Ein- und -Ausgabe bietet Micro-Prolog alle Möglichkeiten, die vom Betriebssystem zugelassen werden. So kann auf jedes Byte einer Datei unmittelbar zugegriffen werden (Random-Dateien) und es gibt Routinen für formatierte Ein- und Ausgabe. Für die Korrektur von Programmteilen – normalerweise von einzelnen Assertionen – gibt es einen Zeileneditor, der allerdings etwas unhandlich ist. Ein Bildschirmeditor, der sich an verschiedene Hardware-Umgebungen anpassen läßt (wie bei Turbo-Pascal), wäre hier sicher die professionellere Lösung.

Ansonsten kann man Micro-Prolog als durchaus professionelles Hilfsmittel für die Entwicklung von KI-Programmen ansehen. Was die Möglichkeiten der deskriptiven Programmierung angeht, steht es Standard-Prolog nicht nach und bietet auch, wie erwähnt, eine ganze Reihe von Werkzeugen für die Programm-Entwicklung. Auf MS-DOS-Rechnern sollte der Speicherplatz auch für echte Anwendungen ausreichen. Zu diesen Anwendungen zählen keineswegs nur Expertensysteme, wie das hervorragende Buch von Clark und McCabe [3], den Erfindern von Micro-Prolog, zeigt; es lassen sich auch Hilfsmittel für das Projekt-Management oder Spielprogramme entwickeln und auch einen Einsatz im Unterricht (etwa Geographie, Geschichte, Biologie) kann man sich gut vorstellen.

```
Diagnose (.) if
      PP and
      X Fehler-Liste and
      (forall mgl-Ursache (X Y Z) then Print-Diagnose (Y Z)) and
      PP

X Fehler-Liste if
      P (  Welche Teile funktionieren nicht?  ) and
      R (X) and
      PP
Print-Diagnose (X Y) if
      P (X    verursacht Fehler an    Y) and
      PP

mgl-Ursache (X Y Z) if
      Y wirkt-auf Z and
      (forall x ON Z then x ON X)

(Dynamo defekt) wirkt-auf (Scheinwerfer Ruecklicht)
(X defekt) wirkt-auf (X) if
      X CON
X wirkt-auf (Y) if
      Z TEIL-VON Y and
      X wirkt-auf (Z)


Diagnose.

  Welche Teile funktionieren nicht?   (Scheinwerfer Ruecklicht Reifen)

(Dynamo defekt)    verursacht Fehler an    (Scheinwerfer Ruecklicht)
(Birne defekt)    verursacht Fehler an    (Scheinwerfer)
(Gehaeuse defekt)    verursacht Fehler an    (Scheinwerfer)
(Birne defekt)    verursacht Fehler an    (Ruecklicht)
(Gehaeuse defekt)    verursacht Fehler an    (Ruecklicht)
(Schlauch defekt)    verursacht Fehler an    (Reifen)
(Ventil defekt)    verursacht Fehler an    (Reifen)
(Mantel defekt)    verursacht Fehler an    (Reifen)
```

Bild 10. Einige Relationen, die zusammen mit der Datenbasis aus Bild 2 einen Experten für Fahrrad-Defekte entstehen lassen, und ein exemplarischer Dialog

Die Dokumentation zu Micro-Prolog umfaßt das erwähnte Lehrbuch [3], das auch separat im Buchhandel erhältlich ist, und ein Referenz-Handbuch, das die Implementation im Detail beschreibt.

Ein rudimentäres Expertensystem

Zum Abschluß ist in *Bild 10* noch ein „Programm" wiedergegeben, das erste Schritte zu einer Anwendung der Fahrrad-Datenbasis zeigt. Es sei hier speziell auf die Pseudorelationen R, P und PP hingewiesen, die imperativen Charakter besitzen. Sie geben keine Beziehung zwischen ihren Argumenten wieder, sondern dienen der Bildschirm-Ein- bzw. -Ausgabe.

An diesem Programm läßt sich noch eine wichtige Eigenschaft „intelligenter" Systeme demonstrieren: Ein solches System weiß praktisch immer eine Antwort, auch wenn die Wissensbasis schlecht oder unvollständig ist – wie im vorliegenden Fall. Dann sind eben auch diese Antworten schlecht oder unvollständig. Eine Verbesserung kann durch Eingabe weiteren genaueren Wissens leicht erzielt werden – das man natürlich erst einmal haben muß.

Literatur

[1] *W. F. Clocksin, C. S. Mellish:* Programming in Prolog, Springer-Verlag Berlin Heidelberg New York 1981.

[2] *G. Schoffa:* Lisp, eine Sprache für die künstliche Intelligenz, mc 9/1985, Seite 50.

[3] *K. L. Clark, F. G. McCabe:* micro-prolog: Programming in Logic, Prentice Hall, Englewood Cliffs, New Jersey 1984.

Werner Stehling

Interrupts bei Prozessoren 680XX

Teil 1: Die Regeln von der Ausnahme

In der guten alten 8-Bit-Welt kam ein Prozessor bequem mit drei verschiedenen Ausnahmebedingungen über die Runden – und schon damit waren die Programme nicht immer leicht zu verstehen. Mit der neuen 16-Bit-Generation ist es wahrhaftig nicht einfacher geworden. Darum zeigt mc, wie die Ausnahmeverarbeitung bei Prozessoren der Serie 68000 gehandhabt wird.

Die erste und wichtigste der Ausnahme-Bedingungen bei 8-Bit-Prozessoren war natürlich der Reset: Hier durfte sich das System mehr oder weniger neu initialisieren. Ein Interrupt, die zweite Ausnahme, sorgte dafür, daß der Prozessor während der Programm-Abarbeitung unterbrochen wurde, um schnell eine andere Tätigkeit auszuführen, und zum dritten konnte so eine Unterbrechung auch per Befehl herbeigeführt werden, ein sogenannter Break.

Die 16-Bit-Welt

Immer getreu der Devise „nicht kleckern sondern klotzen" wird der geneigte Benutzer beim 68000 gebeten, nicht nur drei, sondern 230 verschiedene Ausnahmen zu verwalten. Zum Reset kam der Busfehler dazu, aus einem Interrupt wurden sieben, die man aber auf genau 201 Arten behandeln kann, der Break mauserte sich zu 16 verschiedenen Trap-Befehlen und zusätzlich wurde eine Handvoll Ausnahmen erfunden, auf die man früher bequem verzichten konnte. Bei der ganzen Sache gibt es nur einen Trost: alle Ausnahmen werden im Prinzip auf die gleiche Art behandelt, hat man eine verstanden, dann hat man alle verstanden.
Die Mikroprozessor-Familie der Serie 68000 besteht zur Zeit aus den fünf Mitgliedern 68008, 68000, 68010, 68012 und 68020, und alle Welt behauptet, die Prozessoren seien untereinander aufwärtskompatibel. Nun ist erstens „Kompatibilität" ein sehr dehnbarer (und leider auch arg strapazierter) Begriff und zweitens hat es keinen großen Sinn, die CPU isoliert zu betrachten. Die beiden interessanten Fragen sind deshalb, wie stark sich die Prozessoren in ihrem äußeren Auftreten, also im Zusammenspiel mit der übrigen Hardware, unterscheiden und auf welche Fallstricke der Programmierer gefaßt sein muß, wenn er ein anderes Familienmitglied einsetzen will.
Im Sinne dieser Untersuchung kann man sich auf die Typen 68000, 68010 und 68020 beschränken. Als kleinere Ausgabe des 68000 besitzt der 68008 einen nur acht Bit breiten Datenbus, doch beide verhalten sich bei der Ausnahmeverarbeitung völlig gleich. Die CPU 68012 unterscheidet sich von der 68010 ausschließlich durch einen breiteren Adreßbus und muß daher auch nicht separat betrachtet werden. Wie es sich für eine anständige Familie gehört, sind die Breiten von Adreß- und Datenbus für das Verhalten bei Ausnahmen bedeutungslos, schließlich ist es niemandem verwehrt, den 68020 als 8-Bit-Prozessor zu verwenden.
Im Laufe dieses Beitrages wird, ohne die allgemeine Betrachtung der Ausnahmeverarbeitung zu vernachlässigen, etwas detaillierter auf die Verhältnisse bei Interrupts eingegangen werden. Man kann die Philosophie vertreten, daß alle anderen Ausnahmen nur abgemagerte Interrupts sind, deren Quelle sich meist innerhalb der CPU befindet, eben deshalb ist bei Interrupts nur der Aufwand an Hardware und Verwaltungsarbeit ein bißchen größer. In einem hardwarenahen Teil wird dann auf das Bus-Protokoll eingegangen: wie sieht die Signalfolge auf Adreß-, Daten und Kontrollbus aus, welche CPU-Signale sind überhaupt beteiligt, welche Eigenschaften muß die Peripherie aufweisen. Unter programmtechnischen Aspekten schließlich werden die verschiedenen Stackformate beschrieben, was mit den CPU-Registern geschieht und was man als Programmierer möglichst tun oder lassen sollte. Besonders spannend sind dann die Überlegungen zu Mehrfach-Interrupts mit verschiedenen oder gleichen Prioritäten.

Was ist Ausnahmeverarbeitung

Wie sieht das Leben eines geplagten Mikroprozessors in einem Rechner aus? Er mühlt vor sich hin, holt Befehle und Daten aus dem Speicher, verwurschtelt sie auf irgendeine Weise (oder auch nicht) und speichert das Resultat seiner Bemühungen wieder irgendwo ab. Doch plötzlich passiert etwas Besonderes: Ein Adreß-Strobe tappt ins Leere, unter der gesuchten Adresse meldet sich niemand – wo ist denn nur der Speicher geblieben? Jedenfalls ein völlig neuer Aspekt, wie soll man denn da mit dem normalen Programm fortfahren? Jeder Programmierer hat wohl schon diese trübe Erfahrung gemacht: Die meisten Rechner melden sich dann mit einem häßlichen Piepser und einer Meldung wie: „Ähm – Bus-Fehler – könnten Sie vielleicht mal Reset drücken?"
Wo hier die Ausnahmeverarbeitung steckt? Nun, der Rechner muß irgendwie merken, daß er versucht hat, auf eine nicht existierende Adresse zuzugreifen (Busfehler). Sowie er das festgestellt hat, kann er natürlich nicht im Programm fortfahren, als ob nichts geschehen wäre. Der Vater (oder die Mutter) des Betriebssystems hat aber für diesen Fall vorgesorgt: Das laufende Programm wird abgebrochen, die besagte informative Nachricht erscheint auf dem Bildschirm und die ganze bisherige Rechenzeit ist für die Katz. Das ist Ausnahmeverarbeitung.

Unterschiedliche Arten

Wie man sich denken kann, ist ein Busfehler nur ein möglicher Anlaß, ein Programm nicht normal weiterzuführen. Es können noch andere Fehlerbedingungen auftreten, die vom Programmierer ursprünglich nicht beabsichtigt waren und gesondert behandelt werden müssen. In-

terne Fehler sind aber nicht die einzigen Ursachen für eine Ausnahme. Es gibt durchaus gewollte Unterbrechungen des Programmablaufs, dem Normalverbraucher weithin als Interrupts bekannt. Dabei wird der Prozessor kurzfristig mit einer anderen Aufgabe betraut und führt erst danach das unterbrochene Programm weiter aus. Beispiele sind die Reaktion auf eine kleine Anfrage von einem Peripheriegerät oder -baustein (eine Taste wurde gedrückt, der Drucker kann ein neues Zeichen annehmen) oder ein Timer unterbricht nach einer gewissen Zeit, damit ein anderes Programm auch mal darf (Multitasking).
Die Einteilung in gewollte und nichtgewollte Unterbrechungen ist natürlich völlig willkürlich. Ebenso kann man z. B. zwischen periodischen (Timer, Einzelschritt) und eher zufälligen Ereignissen (Tastatur, Adreßfehler) unterscheiden. Oder wie wäre es mit einer Klassifizierung nach software-generierten (Division durch Null, Einzelschritt) und hardware-korrelierten Unterbrechungen (Busfehler, Timer)? Wie man sieht, sind die Übergänge fließend. Schließen wir uns also lieber Motorolas Standpunkt an und ordnen die verschiedenen Ausnahmebedingungen nach ihrer Wichtigkeit. Aufgrund der Reaktion der CPU auf eine Ausnahmebedingung werden bei 68000 und 68010 drei Gruppen unterschieden:

(0) Die Ausführung des gerade laufenden Befehls ist nicht möglich oder sinnlos (wie etwa beim Busfehler),

(1) der aktuelle Befehl wird fertig abgearbeitet, dann erst beginnt die Ausnahmebehandlung oder

(2) der laufende Befehl selbst führt die Ausnahmebedingung herbei.

Ausnahmen bei 68000 und 68010

Die Ausnahmefälle, die die Prozessoren dabei im einzelnen bearbeiten können, werden im Folgenden nach diesen Gruppen geordnet vorgestellt.

Gruppe 0

0.1 Reset
Ein externes Reset-Signal unterbricht alle laufenden Aktivitäten. Die CPU initialisiert sich neu, indem sie von Adresse $0 des Supervisor-Programm-Bereiches den Startwert des Supervisor-Stackpointers (SSP) und von Adresse $4 den Anfang der Initialisierungs-Routine in den Programmzähler (PC) lädt (alle unverständlichen Ausdrücke werden weiter unten erklärt).

0.2 Address error
Dieser Fehler tritt auf, wenn der Prozessor versucht, einen Befehl oder einen Operanden mit Wort- oder Langwortadressierung auf einer ungeraden Adresse zu erreichen. Die Familien-Philosophie verlangt, daß Worte oder Langworte immer auf geraden Adressen beginnen. In der Ausnahmeverarbeitung kann dann natürlich das Laden von ungeraden Adressen emuliert werden. Der 68020 beschwert sich nur, wenn er einen Befehl auf einer ungeraden Adresse lesen soll, für das Laden von Daten ist er schlau genug.

0.3 Bus error
Ein Busfehler wird außerhalb der CPU durch externe Hardware erkannt. Der Prozessor arbeitet mit allen weiteren Bausteinen des Systems asynchron zusammen, d. h. er legt eine Adresse auf den Adreßbus und erwartet, daß sich innerhalb kürzester Frist irgend jemand angesprochen fühlt. Wenn nicht, erwartet er wenigstens das Bus-Error-Signal.

Gruppe 1

1.1 Trace
Es besteht die Möglichkeit, ein Programm Schritt für Schritt ablaufen zu lassen. Nach jedem Befehl wird zu einer Ausnahmeverarbeitung verzweigt, in der zum Beispiel die Inhalte der Register angezeigt werden können.

1.2...1.8 Interrupt Level 1...7
Externe Geräte oder Bausteine können die Aufmerksamkeit der CPU erheischen und verlangen, daß ihren Bedürfnissen Rechnung getragen wird. So kann etwa ein neuer Meßwert abgeholt oder die Uhrzeit auf dem Bildschirm eingeblendet werden.

1.9 Illegal/unimplemented instruction
Die CPU sieht sich genötigt, einen Befehl ausführen zu sollen, mit dem sie einfach nichts anfangen kann. Von den 65536 möglichen Bitkombinationen gibt es nämlich wahrhaftig noch ein paar, die der Prozessor nicht versteht. Innerhalb der Familie ist die Intelligenz dabei zunehmend: Der 68010 versteht ein bißchen mehr als der 68000, der 68020 noch mehr und was dann noch übrig bleibt, könnte immerhin noch einen Gleitkomma-Coprozessor (Floating Point Unit = FPU) oder eine Speicherverwaltung (Memory Management Unit = MMU) zu hektischer Aktivität veranlassen. Wenn die CPU eine Bitkombination nicht versteht und wenn sich auch kein anderer Baustein im System angesprochen fühlt (etwa weil keine FPU im Sokkel steckt), kann die gewünschte Reaktion immer noch durch die Ausnahmeverarbeitung emuliert werden.

1.10 Privilege violation
Eine Privileg-Verletzung tritt auf, wenn sich der Prozessor in der Anwender-Betriebsart befindet und ein Befehl ausgeführt werden soll, der ausschließlich dem Supervisor (also normalerweise dem Betriebssystem) vorbehalten ist – auch bei Mikroprozessoren funktioniert der Betrieb nur, wenn einer den Daumen drauf hat...

Gruppe 2

2.1 Trap
Die Ausnahmeverarbeitung wird durch einen normalen Befehl eingeleitet. Hauptanwendung ist die Anforderung von Betriebssystem-Aktivitäten durch ein Anwenderprogramm.

2.2 Trapv
Während der Programmausführung kann bei Rechenoperationen ein zu großer Zahlenwert auftreten. Der Test mit einem Trapv-Befehl (neudeutsch: arithmetischer Überfluß) führt zur Ausnahmeverarbeitung.

2.3 CHK
Falls beim Test der Wert eines Registers kleiner als Null oder größer als eine obere Grenze ist, wird eine Ausnahmeverarbeitung eingeleitet.

2.4 Zero divide
Division durch Null gibt halt irgendwie immer Anlaß zum Nachdenken.

2.5 Format error
Dieser Fehler wird dem 68000 nie auffallen. 68010 und 68020 sind da etwas schlauer: sie testen am Ende einer Ausnahmeverarbeitung (beim RTE-Befehl), ob die auf dem Stack abgelegten Daten ihren hohen Ansprüchen genügen. Dieser Test ist zum Beispiel nützlich, wenn der Programmierer in einer Interrupt-Routine schlimme Dinge mit dem Stack anstellt. Der 68020 kennt selbstverständlich noch einige weitere Möglichkeiten, an irgendeinem Datenformat herumzunörgeln, speziell im Zusammenspiel mit Coprozessoren.
Innerhalb der Gruppen 0 und 1 sind die Ausnahmen nach ihrer Wichtigkeit geordnet: Bei gleichzeitigem Auftreten von zwei Ausnahmebedingungen kommt der

zuerst an die Reihe, der weiter oben steht. Die Ausnahmen von Gruppe 2 treten immer im Zusammenhang mit der Ausführung bestimmter Befehle auf und brauchen daher keinerlei Vorrang untereinander. Abschließend ist noch festzuhalten, daß die Ausnahmen der Gruppe 0 die höchste und die der Gruppe 2 die niedrigste Priorität besitzen.

Ausnahmen beim 68020

Der 68020 unterscheidet sogar fünf verschiedene Gruppen. Gegenüber seinen Vorgängern ist die Prioritäts-Reihenfolge etwas abgeändert. Ohne näher darauf einzugehen, hier nur kurz die Gruppeneinteilung:

0.0 Reset
1.0 Adreßfehler
1.1 Busfehler
2.0 Breakpoints, CHK's, CALLM, Traps, Division durch Null u.a.
3.0 illegale und nicht implementierte Befehle, Privileg-Verletzung u.a.
4.0 einige Coprozessor-Befehle
4.1 Trace
4.2 Interrupts

Um es noch einmal ganz klar zu sagen: Die Priorität der Ausnahmebedingung sagt nur etwas darüber aus, welche Ausnahme beim gleichzeitigen Auftreten mehrerer Bedingungen als erste akzeptiert wird. Die Ausnahmeverarbeitungsroutinen werden dann in umgekehrter (!) Reihenfolge abgearbeitet.
Als Beispiel soll hier das Ausführen eines Trap-Befehls im Trace-Modus bei Anliegen eines Interrupts dienen. Als erstes wird in die Trap-Ausnahme verzweigt. Bevor die Verarbeitung jedoch überhaupt losgeht, wird die Trace-Bedingung akzeptiert und in die zugehörige Ausnahmeverarbeitung gesprungen. Doch auch die tut vorläufig einmal nichts, denn nun kommt der Interrupt an die Reihe, der vollständig bearbeitet wird (vorausgesetzt, er schafft das ohne weitere Unterbrechung). Dann erfolgt der Rücksprung in die Trace-Verarbeitung und erst, wenn diese beendet ist, wird die Trap-Bedingung wirklich behandelt. Das gilt natürlich nicht beim Anliegen eines externen Reset-Signals, denn dabei werden alle anderen Ausnahmebedingungen gelöscht. Zwei oder mehr gleichzeitige Interrupt-Anforderungen werden ebenfalls etwas anders behandelt.
Dieses Beispiel funktioniert übrigens nur beim 68020, aber es zeigt das Prinzip. Wie später noch ausführlich beschrieben, stellen die übrigen Prozessoren jedes Tracing erst einmal ab, sobald sie irgend eine Ausnahme akzeptiert haben.

Beginn einer Ausnahmeverarbeitung

Reset, Busfehler und Interrupts (sowie Coprozessoren in 68020-Systemen) sind die einzigen externen Quellen, die eine Ausnahmeverarbeitung anfordern können. Alle anderen Ursachen werden innerhalb der CPU erkannt. Die Behandlung ist jedoch für alle Ausnahmen prinzipiell gleich: Der Prozessor verzweigt sofort in die zugehörige Ausnahmeverarbeitung. Nur Interrupts müssen unter Umständen ein wenig warten, wenn zur Zeit gerade ein anderer Interrupt verarbeitet wird. Bei einer akzeptierten Unterbrechung der Gruppe Null (Gruppe Null oder Eins beim 68020) wird der gerade bearbeitete Befehl abgebrochen, in allen anderen Fällen wird die Instruktion beendet und erst dann in die Ausnahmeverarbeitung verzweigt.
Der Prozessor schaltet dazu in die Supervisor-Betriebsart und rettet gewisse interne Informationen auf dem Stack. Anzahl und Reihenfolge dieser Daten sind einerseits vom vorliegenden Prozessor (68000 oder 68010/20) und andererseits von der Ausnahme selbst abhängig, darauf wird später noch näher eingegangen. In diesem Zusammenhang sollte nochmal daran erinnert werden, daß die Prozessoren zwei Betriebsarten kennen: Den Anwender-Modus, in dem gewisse Befehle verboten sind und sofort zu einer Ausnahmeverarbeitung führen (Privileg-Verletzung), und den Supervisor-Modus, in dem einfach Alles erlaubt ist.
Jeder Ausnahmebedingung ist eine Nummer zugeordnet, die sogenannte Vektornummer. Im allgemeinen wird diese Nummer innerhalb des Prozessors erzeugt, nur bei Interrupts und von Coprozessoren kann sie auch von der anfordernden Peripherie kommen. Die CPU verwendet diese Zahl (nach der Multiplikation mit vier) als Index für eine Tabelle, die die „Ausnahmevektoren" enthält, und lädt die dort gefundene Adresse direkt in den Programmzähler. Dadurch erfolgt ein Sprung zu einer Routine, die dann die eigentliche Ausnahmeverarbeitung vornimmt.

Die Ausnahmevektoren

Der erste Unterschied zwischen 68000 einerseits und 68010/68020 andererseits liegt nun in der Plazierung dieser Vektortabelle im Adreßbereich des Mikroprozessors. Beim 68000 beginnt die Tabelle grundsätzlich bei Adresse $0, die beiden anderen Prozessoren besitzen ein Vektorbasisregister, dessen Inhalt zu der mit vier multiplizierten Vektornummer addiert wird, d. h. bei diesen Prozessoren kann die Vektortabelle (*Tabelle 1*) irgendwo im Adreßbereich untergebracht werden.
Ein Beispiel soll die Verhältnisse verdeutlichen: Nehmen wir wieder den Fall eines Busfehlers. Die zugehörige Vektornummer ist zwei. Der Index für die Tabelle ist somit 4 x 2 = 8 (jeder Adresseintrag belegt 4 Bytes) und der 68000 sucht nun unter der Adresse $8 die Startadresse der Busfehler-Ausnahmeverarbeitungs-Routine (ich liebe diesen Ausdruck!). 68010 und 68020 machen hier nicht so kurzen Prozeß: sie addieren erst noch den Inhalt ihres Vektorbasisregisters dazu, z. B. 8 + 76592 = 76600, und suchen dort nach dem Start der Ausnahmeverarbeitung. Eine Anmerkung am Rande: Der Wert des Vektorbasisregisters muß natürlich gerade sein, sonst hat man zusätzlich auch noch einen Adreßfehler am Hals. Womit auch gleich offensichtlich wird, warum die Adreßfehler-Ausnahme die zweithöchste Priorität besitzt.

Der letzte Befehl einer Ausnahmeverarbeitungs-Routine ist immer ein RTE (Return from Exception). Der Prozessor holt die gleiche Anzahl von Worten vom Stack, die er zu Beginn der Ausnahmeverarbeitung darauf abgelegt hat, und restauriert damit wieder seinen ursprünglichen Zustand. Wenn es sich irgendwie einrichten läßt, sollte er dabei auch einigermaßen sinnvolle Informationen vorfinden. 68010 und 68020 testen diese Daten daher auf Integrität und Plausibilität (wenn man weiß, wie, kann man sie natürlich immer noch hereinlegen).

Interrupts im Detail

Nun soll das spezielle Augenmerk auf die Interrupt-Behandlung gelegt werden. Sie unterscheidet sich von den anderen Ausnahmeverarbeitungen nur in einem Punkt: Die Vektornummer der Ausnahme ist nicht schon von vornherein bekannt, sondern sie wird von der CPU erst in einem zusätzlichen Interrupt-Bestätigungs-Zyklus beschafft.
Eine der wichtigsten Aufgaben eines Mikroprozessors besteht im Reagieren auf äußere Einflüsse. Prinzipiell hat der Prozessor zwei Möglichkeiten, auf die Bedürfnisse der Peripherie einzugehen: Er kann in einer Schleife dauernd ein Gerät nach dem anderen abfragen, ob sich dort etwas tut („polling"), oder er kann dar-

auf vertrauen, daß sich der Baustein schon melden wird, wenn er etwas will – eben per Interrupt. Welche der beiden Methoden im Einzelfall günstiger ist, hängt von der Anwendung und den Umständen ab. Polling ist sicher einfacher zu programmieren und zu überschauen, man wird es einsetzen, wenn der Prozessor ohnehin auf eine Eingabe warten muß und nichts Anderes zu tun hat. Die Verwendung von Interrupts bietet sich vor allem bei seltenen Ereignissen (Tastatur-Eingaben) und bei zeitlich periodischen Vorgängen an. Für ein Multitasking-System ist der Interrupt durch einen Timer-Baustein sogar zwingend notwendig.

Wenn in einem Computer Interrupt-Betrieb möglich ist, dann müssen einige ganz wesentliche Grundregeln beachtet werden:

1. Der Hardware-Entwickler muß dafür sorgen, daß ein Interrupt richtig erkannt und behandelt wird (klingt simpel, ist es aber nicht, wie noch gezeigt wird).

2. Der Programmierer muß dafür sorgen, daß nach dem Interrupt das unterbrochene Programm so weiterläuft, als ob der Interrupt gar nicht stattgefunden hätte. Das bedeutet im Einzelnen, daß alle (!) Register der CPU wieder im ursprünglichen Zustand sein müssen, und es ist allergrößte Vorsicht geboten, wenn eine Interruptroutine in einem Speicherbereich herumfuhrwerkt, der eigentlich einem anderen Programmteil gehört.

3. Wenn ein Interrupt möglich ist, dann ist auch mehr als einer möglich! Diese Aussage klingt so paradox, daß sie der unerfahrener Programmierer gerne übersieht. Die Konsequenz ist jedoch, daß eine Interruptroutine selbst wieder durch eine Ausnahmeverarbeitung mit höherer Priorität unterbrochen werden kann. Und genau hier passieren die häufigsten Fehler, denn es ist (wenn man nicht aufpaßt) relativ einfach, eine Interrupt-Routine zu schreiben, die sich nicht straflos unterbrechen läßt.

Erkennen und Vorbereiten

Im Sinne dieser Beschreibung liegt die Quelle des Interrupts außerhalb der CPU. Deshalb sind (im Gegensatz zu den anderen Ausnahmen) einige zusätzliche Schritte notwendig. Bevor die eigentli-

Tabelle 1:
Vektornummer, relative Adresse (Offset) des Vektors und Ausnahmebedingung

Vektornr. dez.	Vektornr. hex.	Offset dez.	Offset hex.	Ausnahmebedingung
0	00	0	000	Reset, Anfangswert des SSP
	–	4	004	Reset, Anfangswert des PC
2	02	8	008	Busfehler
3	03	12	00C	Adreßfehler
4	04	16	010	ungültiger Befehl
5	05	20	014	Division durch Null
6	06	24	018	CHK-Befehl (Register-Wert testen)
7	07	28	01C	TRAPV-Befehl (auf Overflow testen)
8	08	32	020	Privileg-Verletzung
9	09	36	024	TRACE-Modus (Einzelschritt)
10	0A	40	028	Emulation von $Axxx-Befehlen (%1010...)
11	0B	44	02C	Emulation von $Fxxx-Befehlen (%1111...)
12	0C	48	030	reserviert
13	0D	52	034	Coprozessor-Protokollfehler (68020)
14	0E	56	038	Formatfehler (68010 und 68020)
15	0F	60	03C	nichtinitialisierter Interrupt
16	10	64	040	reserviert
...	...	...	...	reserviert
23	17	92	05C	reserviert
24	18	96	060	Spurious Interrupt (Ausn. + Busfehler)
25	19	100	064	Interrupt-Autovektor Priorität 1
26	1A	104	068	Interrupt-Autovektor Priorität 2
27	1B	108	06C	Interrupt-Autovektor Priorität 3
28	1C	112	070	Interrupt-Autovektor Priorität 4
29	1D	116	074	Interrupt-Autovektor Priorität 5
30	1E	120	078	Interrupt-Autovektor Priorität 6
31	1F	124	07C	Interrupt-Autovektor Priorität 7
32	20	128	080	TRAP-0-Befehl
33	21	132	084	TRAP-1-Befehl
34	22	136	088	TRAP-2-Befehl
35	23	140	08C	TRAP-3-Befehl
36	24	144	090	TRAP-4-Befehl
37	25	148	094	TRAP-5-Befehl
38	26	152	098	TRAP-6-Befehl
39	27	156	09C	TRAP-7-Befehl
40	28	160	0A0	TRAP-8-Befehl
41	29	164	0A4	TRAP-9-Befehl
42	2A	168	0A8	TRAP-10-Befehl
43	2B	172	0AC	TRAP-11-Befehl
44	2C	176	0B0	TRAP-12-Befehl
45	2D	180	0B4	TRAP-13-Befehl
46	2E	184	0B8	TRAP-14-Befehl
47	2F	188	0BC	TRAP-15-Befehl
48	30	192	0C0	reserviert für FPU (68020)
...	...	...	...	reserviert für FPU (68020)
54	36	216	0D8	reserviert für FPU (68020)
55	37	220	0DC	reserviert
56	38	224	0E0	reserviert für MMU (68020)
57	39	228	0E4	reserviert für MMU (68020)
58	3A	232	0E8	reserviert für MMU (68020)
59	3B	236	0EC	reserviert
...	...	...	...	reserviert
63	3F	252	0FC	reserviert
64	40	256	100	192 Interrupt-Vektoren ...
...	...	...	...	...frei ...
255	FF	1020	3FC	... für Anwender.

che Ausnahmeverarbeitung losgeht, läuft im Wechselspiel zwischen Prozessor und beteiligter Hardware folgendes Ritual ab:

- Das anfordernde Gerät teilt der CPU mit, daß es gern bedient werden möchte. Dabei gibt es seine Priorität bekannt, die vom Benutzer vorher festgelegt wurde (in ganz intelligenten Systemen kann diese Einstellung sogar per Software erfolgen). Es kommen die Stufen 1...7 in Frage, wobei 7 die höchste Priorität besitzt. Verschiedene Interrupt-Quellen können durchaus die gleiche Priorität besitzen.

- Der Mikroprozessor vergleicht mit seiner internen Interruptmaske, ob die angemeldete Vorrangstufe hoch genug ist, um das laufende Programm zu unterbrechen. Wenn die Maske etwa die Zahl vier enthält, so werden nur Interrupts der Stufen 5, 6 oder 7 akzeptiert. Die höchste Priorität, also die Stufe sieben, kann nicht maskiert werden, diese Interrupts werden also immer angenommen.

- Die CPU vollzieht dann einen Buszyklus zur Interruptbestätigung, auf dessen Einzelheiten weiter unten eingegangen wird. Das anfordernde Gerät muß nun seinerseits erkennen, daß sein eigener Interrupt behandelt wird und daß die CPU gern wissen möchte, wo die entsprechende Interrupt-Ausnahmebehandlungs-Routine zu finden ist.

- Nachdem der Prozessor nun von der Hardware den heißen Tip bekommen hat, welcher Ausnahmevektor zu verwenden sei, fängt die eigentliche Ausnahmeverarbeitung an.

Beteiligung der CPU

Beim Erkennen und Akzeptieren eines Interrupts sind einige CPU-Register beteiligt: Statusregister, Stackpointer und Vektorbasisregister. Die Außenwelt muß einigen Beinen des Käfers besondere Aufmerksamkeit widmen, speziell den Interrupt-Kontroll-Leitungen, den Prozessor-Status-Signalen, dem Adreßbus und den Kontrollsignalen $\overline{BERR}$ und $\overline{VPA}$. Zwischen den einzelnen Familienmitgliedern bestehen auch hier einige Unterschiede, auf die an den betreffenden Stellen eingegangen wird.
Der Prozessor hat intern die Möglichkeit, einzelne Interrupts zu sperren. Indem er die Interruptmaske auf einen Wert von 0...7 setzt, kann er alle Interrupt-Anforderungen mit einer Priorität kleiner oder gleich der Maske abwimmeln (0 = alles ist erlaubt, 6 oder 7 = nur Priorität 7 kommt durch).
Die Bits 0...7 des Statusregisters (*Bild 1*) enthalten die üblichen Flags für Carry, Overflow, Zero, Negativ und Extend. Sie bilden zusammen das sogenannte „user byte", auf das von allen Anwenderprogrammen aus zugegriffen werden kann. Die hier interessierende Interruptmaske befindet sich in den Bits 8, 9 und 10. Bit 13 (Supervisor-Status) zeigt an, ob sich der Prozessor in der Anwender- (S=0) oder Supervisor-Betriebsart (S=1) befindet. Alle Ausnahmebehandlungen, also auch die Interrupts, werden grundsätzlich im Supervisor-Modus behandelt. Das S-Bit zeigt außerdem noch an, ob der Supervisor- oder der Anwender-Stackpointer in Betrieb ist. Beim 68020 wird zusätzlich im Supervisor-Modus zwischen Master-Stackpointer (M=1) und Interrupt-Stackpointer (M=0) unterschieden.
Bit 15 (Trace-Bit) bei 68000 und 68010 bzw. Bit 15 und 14 bei 68020 steuern die Abarbeitung eines Programms im Einzelschritt-Verfahren. Falls diese Betriebsart eingeschaltet ist, wird nach jedem Befehl eine Einzelschritt-Ausnahmeverarbeitung durchgeführt (unter Verwendung von Vektor Nr. 9, siehe Tabelle 1).

Das obere Byte des Statusregisters (Bit 8...15) ist das sogenannte Systembyte, wie man sieht, haben alle Felder irgend etwas mit Ausnahmeverarbeitung zu tun. Damit nun nicht jedes x-beliebige Anwenderprogramm z. B. die Interruptmaske verändern (wichtig bei Multitasking) oder in den Einzelschritt-Modus schalten kann (wichtig bei mehreren Benutzern), darf das Systembyte nur modifiziert werden, wenn sich der Prozessor in der Supervisor-Betriebsart befindet. Jeder diesbezügliche Versuch im Anwender-Modus führt automatisch zu einer Ausnahmeverarbeitumng über Vektor Nr. 8 (Privileg-Verletzung).

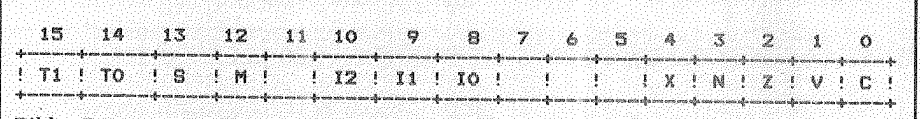


Bild 1. Das Status-Register der Prozessoren 680XX: Im unteren Byte die üblichen Flags, im oberen Byte (Bit 8...15) die Interrupt-Maske, Trace- und Status-Flags. T0 und M sind nur beim 68020 vorhanden

Die Stackpointer

Irgend ein schlauer Mensch hat schon vor langer Zeit durchgesetzt, daß alle gängigen Mikroprozessoren Stackmaschinen zu sein haben. Die Prozessoren der Reihe 680xx sind – noch einen Schritt weiter – typische Hochsprachen-Prozessoren: Ihr Befehlssatz ist an den heute üblichen strukturierten Programmiersprachen orientiert. Darüber hinaus erfolgt eine konsequente Trennung zwischen Supervisor- und Anwender-Betriebsart. Wenn man einmal voraussetzt, daß das Betriebssystem des Computers im Supervisor-Modus läuft und alles andere in der Anwender-Betriebsart, so ist es nur logisch, wenn auch zwei separate Stackpointer vorgesehen sind: Nur dadurch ist es nämlich möglich, ein Betriebssystem zu programmieren, das durch ein Anwenderprogramm unter keinen Umständen zum Absturz gebracht werden kann. Um die ganze Sache noch ein bißchen sicherer (und für den armen Programmierer noch etwas komplexer) zu machen, hat man dem 68020 zusätzlich noch einen eigenen Interrupt-Stackpointer spendiert. Während des Betriebes ist also je nach Zustand der CPU das Register A7 oder A7' (oder A7'' beim 68020) als Stackpointer in Funktion.

Ein Stack ist die einfachste Form dynamischer Speicherplatzverwaltung. Im Zusammenhang mit Interrupts wird er hauptsächlich dazu verwendet, CPU-interne Informationen (Inhalt von Programmzähler, Statusregister usw.) aufzubewahren, damit am Ende der Ausnahmeverarbeitung der ursprüngliche Betriebszustand wieder hergestellt werden kann.

Die Statussignale

Die CPU zeigt der Außenwelt über die drei Funktionscode-Leitungen FC0...FC2 an, von welcher Art der laufende Buszugriff ist (*Tabelle 2*). Die Unterscheidung zwischen Anwenderdaten, -programm, Supervisordaten und Supervisorprogramm kann dazu verwendet werden, den physikalischen Adreßraum des Prozessors zu vervierfachen. Für unsere Betrachungen sind folgende Feststellungen wichtig:

– Alle Ausnahmevektoren (Tabelle 1) außer dem Reset-Vektor (Nr. 0 und 1) liegen im Supervisor-Datenbereich.

– Der Resetvektor liegt im Supervisor-Programmbereich.

– Der Stack für Ausnahmeverarbeitung liegt im Supervisor-Datenbereich.

Von besonderem Interesse für Interrupts ist der CPU-Bereich, der beim 68000 nur „interrupt acknowledge" heißt und auch nur während der Interrupt-Bestätigung auftritt. Bei 68010 und 68020 kommt dieser Code zusätzlich bei der Breakpoint-Bestätigung vor, und der 68020 benutzt diesen Bereich außerdem für die Steuerung einer Speicherverwaltung (MMU) und die Kommunikation mit Coprozessoren. Diese unterschiedlichen Verwendungen des CPU-Bereiches erfordern eine gewisse Aufmerksamkeit beim Hardware-Design eines Mikrocomputers, wenn verschiedene Mitglieder der 680xx-Familie eingesetzt werden sollen.

Interrupt-Ablauf

Wie wird nun die ganze Sache zwischen den beteiligten Parteien abgewickelt – oder mal banal ausgedrückt: Welche Signale geistern durch die Hardware, bis die eigentliche Interrupt-Ausnahmeverarbeitung endlich losgehen kann?
Der Prozessor besitzt drei Eingangsleitungen ($\overline{IPL0}$...$\overline{IPL2}$), über die ein Interrupt angemeldet werden kann. In diesen drei Bit ist gleichzeitig die angeforderte Interrupt-Priorität binär codiert: Liegen alle drei Eingänge auf High-Pegel, so entspricht dies der Priorität 0, es liegt kein Interrupt an. Liegen alle drei Eingänge auf Masse, so enspricht das dem nicht maskierbaren Interrupt (Priorität 7). In den meisten 68000-Mikrocomputern werden diese Eingänge nicht direkt angesteuert, sondern über einen Prioritäts-Encoder (74LS148 oder ähnliche), so daß für die externe Hardware die sieben Prioritätsstufen auf getrennten Leitungen zur Verfügung stehen. So braucht ein Baustein zur Anforderung eines Interrupts nur die seiner Priorität zugeteilte Leitung auf Masse zu ziehen und abzuwarten, was passiert.
Ein beliebter Anlaß für Verwirrung ist übrigens die negative Logik der Interrupt-Eingänge. Um zum Beispiel einen Interrupt der Priorität 4 anzumelden, müssen die drei Leitungen mit „011" beschaltet werden – das sieht aus wie eine binäre Drei. Erst wenn man nun die Bit für Bit invertiert (Einer-Komplement), erhält man „100" und damit die gesuchte Prioritätsstufe 4. Der erwähnte Prioritätsencoder 74LS148 arbeitet ebenfalls in negativer Logik, d. h. wenn etwa sein Eingang Nr. 5 als höchster eine Null aufweist (also die Eingänge 6 und 7 auf Eins liegen, die Eingänge 0...4 sind unwichtig), erscheint am Ausgang „010" und damit wieder genau das Einerkomplement von %101 = 5. Damit sind jetzt hoffentlich alle Klarheiten restlos beseitigt.

Tabelle 2: Bedeutung der Funktions-Codes

Status	FC2	FC1	FC0	Zugriffsart
0	0	0	0	nicht definiert, reserviert
1	0	0	1	Anwender Daten
2	0	1	0	Anwender Programm
3	0	1	1	nicht definiert, reserviert
4	1	0	0	nicht definiert, reserviert
5	1	0	1	Supervisor Daten
6	1	1	0	Supervisor Programm
7	1	1	1	CPU-Bereich

Bei den Interrupteingängen leistet sich der 68008 den einzigen funktionalen Unterschied zum 68000: Da ein Bein eingespart werden mußte, besitzt der 68008 nur zwei „tretbare Schienbeine", nämlich $\overline{IPL0/2}$ und $\overline{IPL1}$. $\overline{IPL0}$ und $\overline{IPL2}$ sind intern miteinander verbunden, und damit versteht der Prozessor nur die Prioritäten 0 (kein Interrupt), 2, 5 und den nicht maskierbaren Interrupt mit Priorität 7.

Interruptbestätigung

Der Prozessor liest periodisch den Zustand seiner Interrupt-Eingangsleitungen, während er sein Mikroprogramm abarbeitet. Wenn der Vergleich mit der Interruptmaske im Statusregister eine erhöhte Priorität oder den Wert 7 anzeigt, so werden von der CPU folgende Schritte ausgeführt:

(1) Der Inhalt des Status-Registers wird intern zwischengespeichert.

(2) Im Statusregister wird das S-Bit gesetzt, d. h. der Prozessor schaltet in die Supervisor-Betriebsart (vgl. Bild 1).

(3) Der Trace-Modus wird abgeschaltet. Während der Ausnahmeverarbeitung gibt es also keine Einzelschritt-Verarbeitung, es sei denn, die Interrupt-Routine selbst schaltet diesen Modus wieder ein.

(4) Die Interrupt-Maske im Statusregister wird auf den aktuellen Wert gebracht, d.h. auf die Prioritätsstufe des gerade akzeptierten Interrupts. Bei einem externen Reset wird die Maske auf 7 gesetzt, womit alle Interrupts außer dem nicht-maskierbaren (NMI) gesperrt werden.

(5) Die CPU bestätigt die Annahme des Interrupts durch eine spezielle Form des Lesezyklus. Die Funktionscode-Leitungen zeigen einen Zugriff auf den CPU-Bereich an (Tabelle 2). Die Adreßleitungen A4...A23 (...A31 beim 68020) sind alle auf 1 gesetzt. Die Adreßleitungen A1...A3 zeigen eine exakte Kopie der neuen Interrupt-Maske des Statusregisters, also das Inverse der Eingangsleitungen $\overline{IPL0}$...$\overline{IPL2}$ und somit die Priorität des akzeptierten Interrupts. Unter dieser Pseudo-Adresse im CPU-Bereich möchte der Prozessor nun dringend die Vektornummer (Tabelle 1) lesen, über die er den Start der Ausnahmeverarbeitungs-Routine finden kann. Der 68020 aktiviert zusätzlich seine $\overline{IPEND}$-Leitung (logisch Null) um anzuzeigen, daß als nächstes eine Interrupt-Bestätigung abläuft.

(6) Nun ist es an der Hardware, dem Informationsdefizit der CPU abzuhelfen. Zuerst einmal muß ihr dazu auffallen, daß ihr eigener Interrupt jetzt behandelt wird, was ja am Zustand der Funktionscode- und Adreßleitungen zu erkennen ist. Hat die Außenstelle gemerkt (oder auch nicht), daß sie gemeint ist, so hat sie drei Möglichkeiten, zu reagieren:

(6.1) Beispielsweise gar nicht, was zweifellos das Bequemste ist. In diesem Fall meldet sich nach einer gewissen Zeit der sowieso (hoffentlich) vorhandene Zeitgeber, z. B. ein Monoflop, und signalisiert einen Busfehler. Im Normalfall reagiert der

Prozessor auf eine Busfehler-Ausnahme mit einem Sprung über Vektor Nr. 2 (Tabelle 1). Tritt der Busfehler jedoch während einer Interrupt-Bestätigung auf, so verwendet die CPU den Vektor Nr. 24 (Spurious Interrupt). Da es jetzt plötzlich niemand gewesen sein will, war der Interrupt offenbar ungewollt (Langenscheidt: spurious = unehelich).

(6.2) Eine etwas höflichere Hardware sollte allerdings der CPU den Gefallen tun, den Pseudo-Lesezyklus ganz normal zu beenden: Sie legt eine Zahl (0...255) auf den Datenbus und aktiviert die $\overline{\text{DTACK}}$-Leitung (oder entsprechend $\overline{\text{DSACK0}}$ und $\overline{\text{DSACK1}}$ beim 68020). Der Prozessor bekommt also seine Vektornummer und alle sind glücklich und zufrieden.

Anmerkung 1:
Es ist unmoralisch, wenn in diesem Fall die Vektornummer kleiner als 64 ist, da die Vektoren 0...63 ja bereits für andere Aufgaben reserviert sind (siehe Tabelle 1), aber man kann natürlich...

Anmerkung 2:
Fragen der Moral verlieren schlagartig jede Bedeutung, wenn es um Geld geht. Eine kleine (selbstverständlich frei erfundene) Geschichte mag das verdeutlichen: Es war einmal eine große Firma, die eine Internationale BüroMaschine unter Verwendung eines INTELligenten Mikroprozessors eines großen Halbleiterherstellers gebaut hatte. Dazu wurde bei einem berühmten Programmschreiber, nennen wir ihn mal MikroWeich, das stolze und erfolgreiche Betriebssystem MW-DOS bestellt. Viele, viele Einheiten wurden an den Mann gebracht, und alle lebten angenehm und sorgenfrei. Nun begab es sich aber, daß der INTELligente Baustein auch einige für zukünftige Entwicklungen reservierte Vektornummern besaß, aber dummerweise machte MW-DOS fleißig Gebrauch von diesen verbotenen Früchten. Als nun einige Zeit später der noch INTELligentere Nachfolgebaustein in großen Stückzahlen auf den Markt geworfen wurde, kam plötzlich Erstaunen auf: Die vom Betriebssystem so eifrig benutzten Vektoren hatten plötzlich andere Aufgaben und waren somit für MW-DOS absolut unverdaulich. Jetzt war guter Rat teuer. Auf der einen Seite eine riesige Anzahl installierter Betriebssysteme, auf der anderen ein entnervter Halbleiter-Hersteller, der andauernd nur so etwas wie „aber das war doch reserviert" vor sich hin murmelte. Das Ende der Geschichte: Ein paar tausend hochINTELligente Bausteine wurden zurückgerufen, denn schließlich kostet die Änderung einer Mikroprozessor-Maske nur ein paar Millionen Dollar, und dessen Kurs sinkt ja ohnehin immer tiefer. Verglichen mit den Kosten für eine Revision von MW-DOS ist das jedenfalls ein Klacks. Was lernen wir daraus? Jeder darf ruhig einen reservierten Vektor verwenden, denn Motorola wird ganz sicher beim Redesign ihres 68030 darauf Rücksicht nehmen. Vorausgesetzt natürlich, man hat schon genügend Kasse mit dem Vektor gemacht...
Doch nun zur dritten Möglichkeit, wie die CPU endlich Ihre Vektornummer finden kann:

(6.3) 68000 und 68010 besitzen einen $\overline{\text{VPA}}$-Eingang (valid peripheral address), der zusammen mit dem $\overline{\text{Enable}}$- und dem $\overline{\text{VMA}}$-Ausgang die Synchronisation mit den altbewährten (und recht betagten) Peripheriebausteinen der Serie 6800 ermöglicht. Während der Interrupt-Bestätigung jedoch besitzt dieser Eingang eine völlig andere Bedeutung: Wenn die Außenstelle die $\overline{\text{VPA}}$-Leitung aktiviert (auf logisch Null), dann merkt die CPU, daß sie vergeblich auf eine Vektornummer von der Peripherie wartet. Sie beendet Ihren Pseudo-Lesezyklus und berechnet die gesuchte Vektornummer direkt aus der Priorität des Interrupts, indem sie zu dieser Priorität 24 addiert (vergleiche Tabelle 1). Der zugehörige Vektor ist dann der sogenannte Interrupt-Autovektor.

Da niemand reinen Gewissens einem 68020 zumuten wollte, mit der guten alten 6800-Peripherie zusammenzuarbeiten, sind bei ihm die entsprechenden Extremitäten im Sinne der Evolution völlig verkümmert: als Ersatz für $\overline{\text{VPA}}$ gibt es nun nur noch einen Autovektor-Eingang $\overline{\text{AVEC}}$, womit wir wieder kompatibel wären.

Zusammenfassung

Reagiert die Außenwelt nicht auf eine Interruptbestätigung der CPU, so erfolgt die Ausnahmeverarbeitung über den Spurious-Interrupt-Vektor. In einem normalen Pseudo-Lesezyklus liest der Prozessor die Vektornummer über den Datenbus, wobei nur die unteren 8 Bit gültig sind. Wird dagegen die $\overline{\text{VPA}}$-Leitung ($\overline{\text{AVEC}}$ beim 68020) aktiv (low), dann startet die Ausnahmeverarbeitung über den Autovektor, der zur Prioritätsstufe des Interrupts gehört.
Die Schritte 5...6 dienen nur dazu, die Interruptquelle zu lokalisieren; sie kommen daher nur bei Interrupts vor. Die vorbereitenden Schritte 1...3 und alle sonstigen Schritte sind nicht interruptspezifisch, sondern werden grundsätzlich bei jeder Ausnahmebehandlung durchlaufen. *(wird fortgesetzt)*

Der Mensch im CIM-Mittelpunkt

Kreativität, Erfindergeist und Engagement des Werkstattmanns können für ein CIM-Konzept wichtiger sein als Maschinenauslastung und Materialausnutzung. Dies ist der Kern des neuen Esprit-Projektes Nr. 1217, das unter dem Namen „Human Centered CIM-System" im Mai 1986 gestartet wurde. Innerhalb von drei Jahren entwickeln und erproben drei nationale Arbeitsgruppen je ein Softwarepaket für CAP, CAM und CAD mit einem Investitionsvolumen von insgesamt 13,2 Millionen DM. In der deutschen Arbeitsgruppe beteiligt sich Krupp Atlas neben der Universität Bremen am CAP-System. Ziel des Vorhabens ist ein flexibles Fertigungssystem für die auftragsorientierte Einzel- und Kleinserienfertigung. Dabei sollen die menschlichen Interventionsmöglichkeiten für Disposition, Überwachung und Steuerung auf Werkstattebene verstärkt genutzt werden. Das Wissen und die Erfahrung des Maschinenbedieners soll in ein CIM-System einfließen, daß auf geänderte Anforderungen und Störfälle schneller reagiert und damit robuster, effizienter und wirtschaftlicher arbeitet. Erstmals kommen in einem Esprit-Projekt Sozialwissenschaftler und Arbeitspsychologen zum Zuge. Neben den fertigungstechnischen Aspekten werden auch arbeitswissenschaftliche Aspekte berücksichtigt. Daraus werden sich Konsequenzen auch für die Berufsausbildung ergeben. An dem Projekt arbeiten neben den deutschen Arbeitsgruppen dänische (CAD) und englische (CAM) mit.

Nach Unterlagen von Krupp

Peter Wollschlaeger

Multitasking und Desk Accessories

Das GEM des Atari ST bietet schon von Haus aus ein eingeschränktes Multitasking. Seine Hauptanwendung sind Desk Accessories. Wenn man weiß, wie es geht, sind diese nützlichen Routinen recht einfach zu programmieren.

Das GEM enthält unter anderem etwas, das man bei großzügiger Betrachtung als Multitasking-System bezeichnen könnte. Damit läuft zwar vieles, was bei einem echten Multitasking üblich ist, nicht, dafür ist es einfach zu programmieren. Immerhin reichen die Multitasking-Fähigkeiten soweit, um, wie wir hier zeigen werden, in einem Editor weiterarbeiten zu können, während ein Text gedruckt wird. Dabei läuft dann noch gleichzeitig eine Digital-Uhr mit, und vier weitere Accessories wie zum Beispiel ein Rechner und ein Terminkalender stehen auf Abruf bereit. Das Betriebssystem kann neben einem Anwenderprogramm bis zu sechs Desk Accessories (praktisch Tasks) verwalten. Außerdem läuft immer der Screen-Manager. Dieser überwacht kontinuierlich die Maus-Aktivitäten außerhalb der Arbeitsfläche des aktiven Fensters und meldet der Task, der dieses Fenster gehört, Events (Ereignisse) wie etwa die Anwahl eines Menüs oder die Bewegung der Rollbalken eines Fensters.

Task-Verwaltung mit zwei Listen

Eine Task hat entweder den Status „bereit" (etwas zu tun) oder „wartend". Wartend ist eine Task, wenn sie auf einen Tastendruck, eine Mausbewegung oder sonst ein externes Ereignis wartet. Zur Kontrolle der Tasks verwaltet GEM zwei Listen.

Die „bereiten" Tasks stehen auf der „Ready List", die anderen auf der Warteliste, die im GEM-Handbuch als „Not Ready List" bezeichnet wird. Damit immer nur eine Task zu einem Zeitpunkt die CPU beanspruchen kann, gibt es einen Dispatcher (Abfertiger), der entscheidet, wann welche Task „dran" ist.

Der Dispatcher macht sich seine Aufgabe recht einfach. Zuerst ist die Task (das Programm) aktiv, die ganz oben auf der „Ready List" steht. Diese Task kann eigentlich laufen, solange sie will; wenn es sein muß, ewig. Als einzige Ausnahme gilt: Ruft die Task eine AES-Routine (Teil des GEM) auf, landet sie automatisch beim Dispatcher. Der unterbricht die Programmausführung und setzt die Task an das Ende der Ready-Liste. Alle anderen Tasks rücken nun um einen Platz auf. Die oberste Task wird jetzt bearbeitet. Dieses Prinzip der Listenverwaltung nennt man „round robin". Ruft eine Task eine AES-Routine auf, die ein Warten auf ein Ereignis (Event) erfordert (z. B. Warten auf einen Tastendruck), dann kommt diese Task auf die Warteliste (Not Ready List). Aus der Ready-Liste wird sie entfernt, und die anderen Tasks rücken auf. Das GEM überwacht ab diesem Zeitpunkt, ob das erwartete Ereignis eingetreten ist. Wenn ja, kommt die Task wieder auf die Ready-Liste, allerdings an deren Ende. Theoretisch hört sich dieser Vorgang kompliziert an. Praktisch ist das Verfahren jedoch einfach. Dazu ein Beispiel: Mit dem Booten des Atari ST wurden einige Desk Accesories installiert, aber keines ist aktiv. Die Programme im Hintergrund warten darauf, aufgerufen zu werden. Sie stehen also alle auf der Warteliste. Das gerade aktive Programm wird wahrscheinlich auch darauf warten, daß Sie die Maus bewegen oder eine Taste drücken; auch diese Task steht auf der Warteliste.

Der Normalfall ist also, daß alle Tasks auf der Warteliste stehen und GEM prüft, ob ein Ereignis für einen Task eingetreten ist. Wenn ja, kommt diese Task sofort auf die Ready-Liste und ist gleich an erster Stelle. Wahrscheinlich führt die Task gleich wieder einen AES-Call aus, etwa der Art, „auf nächste Taste warten", und schon ist sie wieder auf der Warteliste.

So kommt ein Desk Accessory auf die Liste

Für ein Desk Accessory heißt das, daß es zuerst einmal in die Warteliste eingetragen werden muß, damit GEM überhaupt merkt, daß es da ist. Das funktioniert so, daß GEM beim Booten prüft, ob Dateien mit der Extension „ACC" auf der Diskette in Laufwerk A vorhanden sind. Das erste „ACC" wird in den RAM-Bereich geladen und sofort wie ein Programm ausgeführt. Wie es sich für ein richtiges Accessory gehört, wird es zu Anfang eine AES-Funktion aufrufen, die „Warten auf ein Ereignis" auslöst. Damit geht es auf die Warteliste. Danach prüft das GEM, ob es weitere „ACC"-Dateien gibt. Mit diesen wird entsprechend verfahren.

Wenn Sie mal probehalber gegen diese Regeln verstoßen und auf einer Diskette eine TOS-Anwendung (ohne AES-Aufrufe) in „ABC.ACC" umbenennen und damit booten, wird GEM ewig dieses „Accessory" laden, weil es auf der Diskette immer wieder eines findet (und immer dasselbe), das nicht auf seiner Liste steht.

Schreiben Sie hingegen ein GEM-Programm, das nichts weiter tut, als eine AES-Funktion der Art „warte auf Ereigns" aufzurufen, dann können Sie es auf „*.ACC" taufen. Beim Booten gibt es damit keine Probleme. Allerdings erscheint das Programm nicht im Desk-Menü; dafür unter Umständen ungerufen auf dem Bildschirm. Ein richtiges Desk Accessory muß zuerst die Funktion „Menu_Register" aufrufen. Damit kann (muß es aber nicht) sich das Programm selbst in dieser Menü-Leiste eintragen. Nehmen wir an, ein Desk Accessory sei in das System eingebunden und steht nun auf der Warteliste. Das Accessory selbst wartet auf ein Ereignis. GEM überwacht das System, und wenn das Ereignis für dieses Accessory eintritt, wird es aktiviert. Es prüft nun, welches Ereignis aufgetreten ist.

Von den vielen möglichen Ereignissen ist eines der wichtigsten wohl „AC_-Open". Diese AES-Meldung bedeutet, daß eines der Accessories im Desk-Menü angewählt wurde. Nun kann das Programm starten und im Rahmen seiner Aktivitäten auch auf andere Ereignisse warten. Um genau zu sein: Gewartet wird immer auf irgendein Ereignis. Das Programm muß dann selbst feststellen, von welcher

Art das Ereignis ist. Ein Desk Accessory wartet zuerst typischerweise immer auf ein „AC_Open". In vielen Listings sieht man eine weitere Prüfung, nämlich ob die richtige Menü-Zeile (msg[4]=Menu_id) angewählt wurde. Das ist dann der Gürtel zum Hosenträger. Zumindest in ST-Pascal hatte ich damit Probleme, weil die Routine „Menu_Register" manchmal falsche Werte lieferte. Unter Megamax-C gab es diesen Ärger nicht.
Während der Debug-Phase hatte ich ein seltsames Erlebnis. Weil ich versehentlich eine Zeile zuviel auskommentiert hatte, fehlte plötzlich jeglicher Text. Trotzdem lief das Desk Accessory ganz normal, solange das Desktop nicht verlassen wurde. Als ich dann die Megamax-Shell aufrief, meldete sich das Desk Accessory unaufgefordert, arbeitete ganz normal, verschwand auch, als es das sollte, nur dann hing das System.
Warum?
Ist ein Desk Accessory aktiv, muß es natürlich auf alle Ereignisse achten, die es selbst benötigt, und zusätzlich auf ein ganz spezielles.
Es gibt nämlich ein Ereignis des Typs „AC_Close", womit GEM einem Accessory meldet: „Verschwinde, mach aber einen ordentlichen Abgang." Ein ordentlich programmiertes Accessory wird in diesem Falle seine Fenster schließen und den Monitor freigeben. So eine Meldung tritt auf, wenn die Anwendung den Bildschirm löschen will oder das Programm beendet ist (also zum Desktop zurückkehrt). Daß man diese Meldung bei bestimmten Anwendungen ignorieren muß, werden wir noch zeigen.

```
PROGRAM Mein_DA;

PROCEDURE do_it;
BEGIN
 Hier steht alles, das DA tun soll
END;

BEGIN
Initialisierung;
WHILE TRUE DO {endlos}
 BEGIN
    Warte auf Event;
    do_it;
 END
END.
```

Bild 1. Das Prinzip eines Programms, das als Desk Accessory laufen soll

Nach dieser langen Vorrede nun zur Praxis. Die Struktur eines Desk Accessories sieht grundsätzlich so aus, wie es *Bild 1* zeigt.
Mit „while true" wird eine Endlosschleife gebildet. Das heißt nicht, daß das Programm endlos läuft, sondern nur, daß es solange zur Verfügung steht, wie der Computer eingeschaltet ist. Sie erinnern sich: Mit „warte auf Ereignis" übernimmt der Dispatcher die Kontrolle. Tritt ein Ereignis ein, wird das Programm aktiviert, die Folgeanweisung (do_it) wird ausgeführt. Dann geht es über die Schleife wieder zu „warte auf Ereignis". Das Desk Accessory wandert wieder auf die Warteliste, und die Anwendung wird fortgesetzt.
Bild 2 zeigt ein Beispiel in ST-Pascal (Plus-Version). Dieses Accessory soll einen Drucker auf NLQ schalten und die Null mit Querstrich drucken oder ihn wieder in den Ausgangszustand zurücksetzen.
Bei der Initialisierung muß lt. Bild 2 folgendes geschehen: Die Funktion „Init_GEM" liefert (wenn erfolgreich) eine positive Zahl. Das ist die Applikationskennung (ID). Die ID brauchen wir, sowie eine Stringvariable mit dem Namen des Desk Accessories. Diese beiden Parameter werden der Funktion „Menu_Register" übergeben, die dafür sorgt, daß der Name des Desk Accessories im Desk-Menü eingetragen wird.
Beachten Sie bitte, daß der Name mit zwei Leerzeichen beginnen muß. Eine Variable muß es sein, weil „Name" in ST-Pascal ein Var-Parameter ist. Nun startet die schon genannte Endlos-Schleife mit der Get_Event-Funktion. Aus deren Angebot interessiert nur eine Nachricht, nämlich „msg[0]" und davon nur „AC_Open". Wurde unser Desk Accessory gerufen, dann startet der Job (do_it), ansonsten geht die Kontrolle wieder an den Dispatcher.
„Do_it" bringt eine Alarm-Box, wenn der Drucker nicht bereit ist, ansonsten ein Fenster mit den Knöpfen „NLQ" und „Normal". Je nachdem, welcher „Button" per Maus gedrückt wurde, wird nun die eine oder die andere Escape-Sequenz an den Drucker gesendet. Die Codes gelten für Star-Drucker im IBM-Modus.
Bild 3 bringt das gleiche Programm in Megamax-C. Primär soll damit gezeigt werden, daß man Desk Accessories eigentlich in jeder Sprache schreiben kann, wenn nur zwei Voraussetzungen

```
{$S0D-}
PROGRAM nlq;

  CONST
    {$I GEMCONST.PAS}

  TYPE
    {$I gemtype.pas}

  VAR
    ap_id,
    menu_id,
    event,d: integer;
    name:Str255;
    msg:Message_Buffer;

  {$I gemsubs}

procedure do_it;
 var button:integer;

 function prt_ready: integer;
   GEMDOS($11);

 begin
   button:=1;
   while (prt_ready=0) and (button=1) do
    button:=Do_Alert('[1][Drucker nicht bereit][Jetzt OK|Abbruch]',1);
   if button=1 then
   begin
     button:=Do_Alert('[2][Druckereinstellung][NLQ|Normal]',1);
     rewrite(output,'lst:');
     if button=1 then write(chr(27),'4',chr(27),'\1')
                 else write(chr(27),'@');
   end;
 end;

begin
    ap_id:=Init_Gem;
    IF ap_id > 0 THEN
      begin
        name:='  NLQ-Listing';
        menu_id:=Menu_Register(ap_id,name);
        while true do
          begin
            event:=Get_Event(E_Message,0,0,0,0,
                             false,0,0,0,0,false,0,0,0,0,
                             msg,d,d,d,d,d,d);
            if msg[0] = Ac_open then do_it;
          end;
      end ;
end.
```

Bild 2. Ein erstes „Schreibtischutensil" in ST-Pascal: Drucker-Umschaltung mit der Maus

```
#include <gemdefs.h>
#include <osbind.h>

extern int      gl_apid;
int menu_id, msg[8];

char prtcmd[][]= { {27,'4',27,'\\','1',0},
                   {27,'@',0}
                 };

do_it()
{
  int i, button;
  char *cp;

  button=1;
  while((Cprnos()==0) && (button==1))
    button=form_alert(1, "[1][Drucker nicht bereit][Jetzt OK|Abbruch]");
  if (button==1)
    {
      button=form_alert(1, "[2][Druckereinstellung][NLQ|Normal]");
      cp=prtcmd[button-1];
      i=0;
      while(cp[i])
        Cprnout(cp[i++]);
    }
}

main()
{
   appl_init();
   menu_id=menu_register(gl_apid,"  NLQ + 0");

  for(;;) /* for ever */
    {
      evnt_mesag(msg);
      if (msg[0] == AC_OPEN) do_it();
    }
}
```

Bild 3. Das Programm aus Bild 2 in Megamax-C. Das Entwicklungssystem muß Desk Accessories unterstützen

gegeben sind. Zum einen muß die Sprache natürlich AES-Aufrufe erlauben, zum anderen muß eine spezielle Stack-Routine vorhanden sein. Ein Accessory ist speicherresident und muß neben den üblichen Bereichen für Code und Daten auch einen eigenen Stack haben. Deshalb startet ein Desk Accessory mit einer kleinen Routine der Art „Reserviere Stack, rufe Desk Accessory auf". Diese Routine gehört zu einem anständigen Entwicklungssystem für den Atari ST dazu und wird beim Linken zuerst eingebunden. Beim Megamax-C ist die Sache recht einfach, da „linkt" man „ACC.L" und dann seine eigene Objekt-Datei. In ST-Pascal ruft man ein Programm auf, das dieses Modul erzeugt, wobei es zuerst fragt, wie groß der Stack sein soll. In diesem Falle reicht 1 KByte (die kleinste Einheit) voll aus, bei größeren Anwendungen dürfte man mit 5 KByte noch auf der sicheren Seite liegen.

Ein Desk Accessory läuft immer

Ein Desk Accessory läuft immer, und sei es, daß es nur darauf wartet, aktiviert zu werden. Dies kann man nutzen, indem man das „Schreibtisch-Utensil" anstatt nur zu warten etwas Sinnvolleres tun läßt. Als erstes Beispiel dafür habe ich eine Uhr gewählt. Das Programm hat die Aufgabe, die Uhrzeit ganz rechts in der Menü-Leiste ständig anzuzeigen. Bisher habe ich auf eine eigene virtuelle „Work Station" verzichten können, weil die Alarm-Boxen (gemeinsam mit den Menüs) einen eh vorhandenen Puffer benutzen. Jetzt muß eine solche Arbeitsstation geöffnet werden. Die Routine dafür halte ich in der Header-Datei „gembas.h". Deren Inhalt (auf diesen Fall hin gekürzt) zeigt *Bild 4*, das eigentliche Uhr-Programm ist in *Bild 5* zu sehen. Die drei Zeilen ab „appl_init" initialisieren und öffnen die „Work Sta-

```
#include <gemdefs.h>
#include <osbind.h>

extern int gl_apid;

int contrl[12];  /* keep linker happy */
int intin[128];
int ptsin[128];
int ptsout[128];
int intout[128];

int work_in[11]; /* work arrays */
int work_out[57];

int gl_hchar;    /* system sizes */
int gl_wchar;
int gl_wbox;
int gl_hbox;

int handle, phys_handle;

open_vwork()
{
int i;
  for(i=0;i<10;work_in[i++]=1);
  work_in[10]=2;
  handle=phys_handle;
  v_opnvwk(work_in,&handle,work_out);
}
```

Bild 4. „gembas.h": eine Header-Datei, die man fast immer braucht

```
/* Uhr als Accessory */

#include "gembas.h"

main()
{
  long zeit;
  char t, *t_string = "00:00:00";

  appl_init();
  phys_handle=graf_handle(&gl_wchar,&gl_hchar,&gl_wbox,&gl_hbox);
  open_vwork();
  vst_color(handle,1);
  vst_effects(handle,16);

  for(;;)  /* for ever */
  {
    zeit=Gettime();
    t = zeit >> 11 & 0x1f;          /* Stunden  */
    t_string[0] = t / 10 + '0';
    t_string[1] = t % 10 + '0';
    t = zeit >> 5 & 0x3f;           /* Minuten  */
    t_string[3] = t / 10 + '0';
    t_string[4] = t % 10 + '0';
    t = 2*(zeit & 0x1f);            /* Sekunden */
    t_string[6] = t / 10 + '0';
    t_string[7] = t % 10 + '0';
    v_gtext(handle,555,gl_hchar-2,t_string);
    evnt_timer(1900,0);
  }
}
```

Bild 5. Das ist alles, und schon läuft ständig eine Uhr rechts oben im Bild

tion". Danach setzte ich die Zeichen-Farbe auf schwarz und den Text-Stil auf Konturschrift. Mit „for(;;)" starte ich die Endlosschleife (immer „while(1)" ist ja langweilig) und hole dann über einen GEMDOS-Aufruf die System-Zeit.
Sie wird in einem Wort der Form „hhhhhmmmmmmssssss" geliefert. Das Programm maskiert immer die passenden Bit, wandelt sie in ASCII um und schreibt die Zeichen in den String „t_string". Leider sind die Sekunden wie beim IBM-PC nur in 5 Bit codiert, weshalb diese Uhr im 2-Sekunden-Takt läuft und dabei die Werte von 0 bis 29 ausgibt. Diese werden mit zwei multipliziert, womit dann die Sekunden von 0 bis 58 angezeigt werden können. Es gibt nun allerhand Tricks, auch die Zwischenwerte anzuzeigen, genau ist aber in dieser Anwendung keiner, denn man weiß ja nie, wie lange das Desk Accessory auf der Warteliste steht. Damit es überhaupt läuft, ist mindestens ein AES-Aufruf Pflicht.
Hier rufe ich den Timer auf, und lasse das Programm fast zwei Sekunden warten, eher gibt es keine neue Zeit.
Nun ist Ihnen sicherlich aufgefallen, daß dieses Desk Accessory nicht via „menu_register" eingetragen ist. Das muß nicht sein, denn wenn Sie den Atari ST booten, erscheint rechts oben „00:00:00". Diese Anzeige bleibt solange stehen, bis

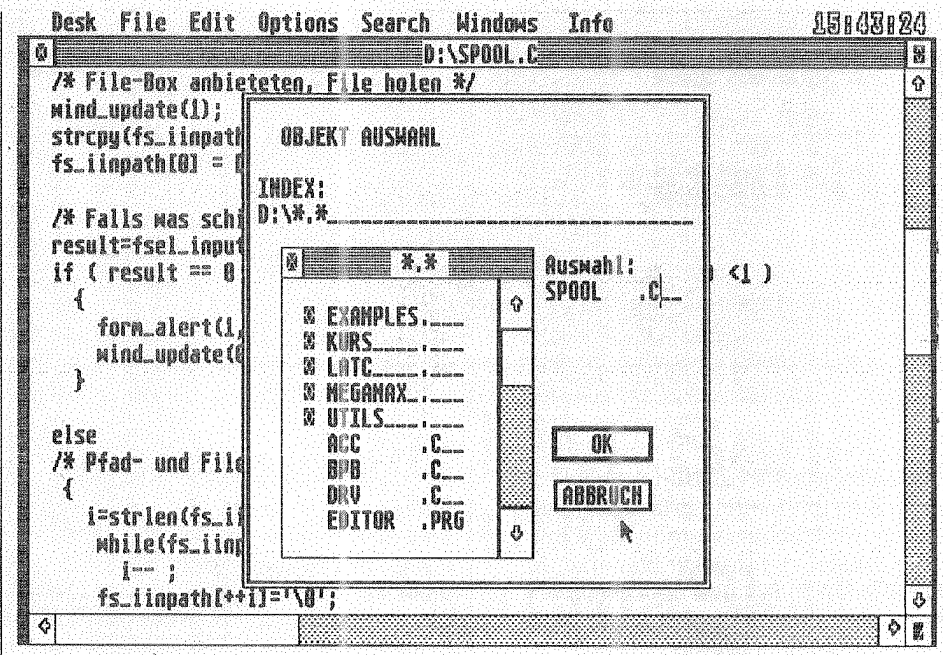


Bild 6. Multitasking: Editor, Uhr und Drucker-Spooler scheinbar gleichzeitig in Aktion

Sie mittels Kontrollfeld die Uhr gestellt haben. Dann erscheint rechts oben dieselbe Zeit und die Uhr läuft. Sie läuft übrigens auch nach einem Reset korrekt weiter. Während einer TOS-Anwendung verschwindet die Uhr, danach erscheint sie wieder, und zwar immer mit der richtigen Zeit.

Spooler ganz einfach

Als letztes möchte ich noch einen Drukkerspooler vorstellen, der im Kern nur aus drei Zeilen besteht. Daß das Programm etwas länger geworden ist, liegt am „Luxus". Wenn Sie das Desk Accessory „Spooler" aufrufen, erscheint eine

```
/* Drucker-Spooler als Desk Accessory
   (c) 1986 Peter Wollschlaeger
*/

#include <stdio.h>
#include <osbind.h>
#include <gemdefs.h>

extern int      gl_apid;
int menu_id, msg[8];

spool()
{
  char fs_iinpath[80], fs_iinsel[80];
  int  *fs_iexbutton, result;
  int fd, i;
  char *buf_ptr;
  unsigned f_len;

  /* File-Box anbieteten, File holen */
  wind_update(1);
  strcpy(fs_iinpath," :\*.*");
  fs_iinpath[0] = Dgetdrv()+'A'; /*aktueller Drive */

  /* Falls was schief ging, Alarm-Box */
  result=fsel_input(fs_iinpath, fs_iinsel, &fs_iexbutton);
  if ( result == 0 || fs_iexbutton == 0 || strlen(fs_iinsel) <1 )
    {
      form_alert(1, "[1][Es wird nicht gedruckt][ OK ]");
      wind_update(0);
    }

  else
  /* Pfad- und Filenamen zusammensetzen */
   {
     i=strlen(fs_iinpath);
     while(fs_iinpath[i] != '\\')
       i-- ;
     fs_iinpath[++i]='\0';
     strcat(fs_iinpath, fs_iinsel);

  /* File oeffnen und in RAM kopieren */
     fd=open(fs_iinpath,0);
     f_len = (unsigned) lseek(fd,0L,2);
     buf_ptr =  malloc(f_len);
     lseek(fd,0L,0);
     read(fd,buf_ptr,f_len);
     close(fd);
  /* nun immer 1 Char drucken, wenn Printer bereit,
     ansonsten via evnt_timer Komtrolle an AES
  */
     wind_update(0);
     i=0;
     while(i < f_len)
      {
       if (Cprnos != 0) Cprnout(buf_ptr[i++]);
       evnt_timer(0,0);
      }
     free(buf_ptr);
   }
}

main()
{
  appl_init();
  menu_id=menu_register(gl_apid,"  Spooler");
  for(;;)/* ever */
   {
     evnt_mesag(msg);
     if (msg[0] == AC_OPEN) spool();
   }
}
```

Bild 7. Ein Drucker-Spooler als Desk Accessory. Der Spooler selbst besteht aus nur drei Zeilen, der Rest ist fast nur „Luxus"

Spruch des Monats

„Jeder Bundesbürger hat über den Computer sein ganz individuelles Meinungsbild; aber nicht jede beliebige Meinung über den Computer läßt sich ernsthaft vertreten."

Aus einer Presseinformation des Software-Serviceunternehmens Cap Gemini

File-Box wie die von *Bild 6*. Sobald Sie dort eine (hoffentlich druckbare) Datei ausgewählt haben, verschwindet die Box wieder, der Drucker läuft und Ihr Programm auch – und die Uhr auch noch, also immerhin schon drei Tasks (scheinbar) gleichzeitig.
Bild 7 zeigt die Lösung dieser Aufgabe. In der Funktion „main" steht nichts Neues, sondern sinngemäß nur wie gehabt „wenn Accessory gerufen wurde, dann spool".
Die Funktion „spool" verschwendet zuerst viel Aufwand auf die File-Select-Box. Diese sollte mit einem Pfadnamen aufgerufen werden, ich setze hier einfach nur „aktueller Drive*.*"ein. Danach sollte in „fs_iinpath" der gewählte Pfad stehen, nur leider hat die Funktion „fssel_input" einen Fehler. Nach dem Pfadnamen steht nämlich noch der alte, also hier „*.*". Bevor ich mich darum kümmere, teste ich zuerst, ob sonst etwas schief ging und gebe ggf. Alarm. Wenn nicht, bereinige ich den Mangel, indem ich rückwärts im Pfadnamen den ersten Backslash suche und danach mit einem Null-Byte das neue String-Ende markiere. An den Pfadnamen wird nun der Datei-Name gehängt und mit diesem Ausdruck die Datei geöffnet. Mit „Iseek" hole ich die Dateigröße, reserviere entsprechend mit „malloc()" Speicher und kopiere die Datei dort hinein.
Erst jetzt folgt der Spooler, und der ist wirklich simpel.

Prinzip:

```
Solange noch ein Zeichen zu drucken
{
wenn Drucker bereit, drucke ein Zeichen
evnt_timer(0,0);
}
```

Mit dem Timer-Aufruf geht die Kontrolle wieder an den Dispatcher, der läßt alle anderen Tasks einmal laufen und dann ist wieder die Druckschleife an der Reihe.
Man merkt schon, daß der Drucker dabei etwas (aber nicht viel) langsamer wird, doch wer will, kann da noch optimieren. Die gängige Lösung ist, in einer weiteren Schleife den Druckerpuffer zu füllen, und dafür den Event-Timer auf eine Zeit zu setzen, die der Drucker benötigt, um seinen Puffer auszudrucken. Damit ist das Programm dann allerdings nicht mehr unabhängig vom Druckertyp.
Dieser Spooler sollte nur eine Anregung sein, korrekte Programmierer werden wohl auch nach dem Open-Befehl prüfen, ob da nicht etwas schief ging und eine Möglichkeit vorsehen, den Druckvorgang abzubrechen. Wenn Sie den Drucker ausschalten, passiert auch nicht viel. Es wird halt nicht gedruckt, aber die anderen Tasks laufen weiter.

Fazit: recht nützlich

So schön das nun alles aussieht, echtes Multitasking ist das nicht, was man wohl am deutlichsten daran sieht, daß eine neue Task nur über die Reset-Taste installiert werden kann, und dann aktiv bleibt, bis man wieder zur Reset-Taste greift.
Andererseits: was man als typischer „Single User" so braucht, läßt sich sehr einfach programmieren, sogar in Hochsprachen. Wenn ich mich so zurückerinnere, wie ich einst in Assembler Interrupt-Vektoren verbogen und meinen eigenen Task-Dispatcher geschrieben habe, so sehe ich hier doch einen deutlichen Fortschritt.

Maus mit Schlitzaugen

Mäuse gibt's wie Sand am Meer. Die Wahl fällt dem geplagten Interessenten schwer. Wonach soll er sich richten? Daß die angebotenen Produkte ihre Aufgabe, nämlich dem Computer punktgenau eine Position mitzuteilen, erfüllen, davon kann man wohl ausgehen. Wir haben uns die NCE-Hyper-Mouse für IBM-PC-kompatible Systeme näher angesehen, die von der Firma Nordphon Computer Electronic, Tarp, angeboten wird.
Selten war ein Testgerät so schnell am PC angeschlossen wie die Hyper-Mouse. Sie verfügt nämlich über einen V.24-Stecker, über den sie auch mit Strom versorgt wird. Eine Einsteckkarte ist nicht erforderlich. Kaum jemals war ein Gerät auch so schnell getestet: Die beiliegende Diskette einlegen, Treiberprogramm starten, Testprogramm starten, Cursor mit Maus positionieren – und fertig. Wie nicht anders zu erwarten, tat das Gerätchen genau das, was man von ihm erwartet hatte.
Der Rest ist schnell erzählt: Die Hyper-Mouse verhält sich wie ihre weit berühmtere Kollegin von Microsoft. Will man ein fertiges Programm mit ihr „bedienen", dann setzt man den Treiber einfach an die Stelle des Microsoft-Treibers. Wir haben das mit einem verbreiteten Grafikprogramm getan, und auch hierbei gab es keine Schwierigkeiten.
Die Hyper-Mouse kommt aus japanischer Fertigung und kostet 298 DM. Die Treibersoftware wurde bei NCE selbst erstellt. Sollten wider erwarten Probleme auftauchen, kann man sich direkt an die verantwortlichen Leute wenden. Ob man allerdings Auskunft bekommt, steht auf einem anderen Blatt. Uns ist es bisher nicht gelungen, von der Firma zu erfahren, wie die Maus von Turbo-Pascal aus angesprochen wird. Aber der findige Computer-Freak ist es ja gewohnt, sich solche Informationen zu beschaffen, ohne dem Anbieter auf den Wecker zu fallen.
Bleibt noch ein letzter Punkt: die Verarbeitung. Auch hier kann man der Hyper-Mouse nichts Schlechtes nachsagen. Sie macht innerlich wie äußerlich einen guten Eindruck. Der Aufbau ist denkbar einfach: Die mit Silikon beschichtete Rollkugel wird von einer Ecke her mit einer gefederten Rolle an zwei Metallwellen angedrückt, die im rechten Winkel zueinander angeordnet sind. Die beiden Wellen treiben Lochscheiben an, die optisch abgetastet werden. Zwei ICs sorgen für die Aufbereitung der Signale. Den Rest übernimmt die Software.

Rudolf Hofer

Heike und Manfred Fillinger

Videodat-Empfänger für den C-64

Empfang von Daten aus dem Fernsehbild

Videodat ist ein Verfahren zur Übertragung von Daten (Texte, Programme ...) über das Fernsehbild – ohne dieses merklich zu stören. Der Zuschauer kann diese Informationen nutzen, indem er den Videoausgang seines Fernsehgerätes oder seines Videorecorders über ein preiswertes Interface mit der seriellen Schnittstelle seines Computers verbindet. Als Empfangs-Software genügt ein gutes DFÜ-Programm.

Neben Videotext und VPS ist Videodat eine weitere Methode zur Übertragung von Dateninformationen im Fernsehbild. Dieses Verfahren wurde vom WDR-Computerclub entwickelt und löste dort die Übertragung von Daten über den Tonkanal ab.
Videodat-Informationen werden zur Zeit in folgenden WDR-Sendungen angeboten:

– WDR Computerclub (auch im HR und ab 1987 auch NDR, RB, SFB)

– Computershop

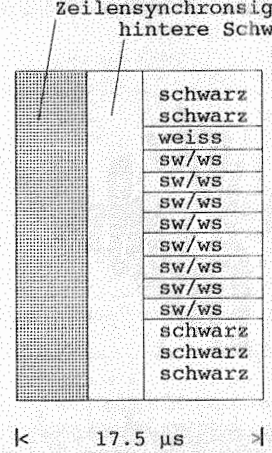
– Schulfernsehen „Bit & Bytes"

Wichtige Begleitinformationen zur Sendung werden dabei mit Videodat übermittelt. Bei der Sendung Computerclub werden neben Informationen auch eine Fülle interessanter Computerprogramme an den Zuschauer weitergegeben.
Die Übertragungsgeschwindigkeit beträgt 300 Baud. In 30 Minuten Sendezeit können ungefähr 20 ziemlich vollgeschriebene DIN-A4-Seiten übertragen werden.
Man kann daher Videodat-Anwendungen für solche Sendungen einsetzen, die dem Zuschauer eine Menge an Informationen – wie zum Beispiel Nachbauanleitungen, Beschreibungen, Testergebnisse... – anbieten. Zur Zeit kann man solch ausführliches Begleitmaterial zu den Sendungen nur über den Postweg bekommen, wenn Videodat nicht genutzt wird. Die Videodat-Übertragung ist durch eine kleine, kaum erkennbare schwarze Fläche in der oberen linken Ecke des Fernsehbildes gekennzeichnet. Während der Datenübertragung werden in dieser Fläche bestimmte Zeilen hell getastet. Dies geschieht nach dem asynchronen, seriellen Datenübertragungsverfahren. Das Startbit wird durch die 25. Zeile (bzw. Zeile 338) markiert und ist weiß. In den folgenden zehn Zeilen werden acht Datenbits und zwei Stopbits ausgestrahlt (siehe *Bild 1*).
Diese Anordnung und Belegung der Zeilen wurde so gewählt, daß auf der einen Seite der Bildinhalt während der Übertragung nur unwesentlich gestört wird und auf der anderen Seite die Rückgewinnung der Daten mit einem einfachen Interface erfolgen kann.
Das Interface „Videodat 300" ist vom anzuschließenden Computertyp unabhängig. Informationen über Videodat und eine Bauanleitung können Sie bei der Computerclub-Redaktion gegen Einsendung eines frankierten Rückumschlages beziehen (siehe auch [1]).
Der Eingang des Interface wird mit dem Videoausgang des Fernsehgerätes oder des Videorecorders (denn Videodat ist „videorecorderfest") verbunden (*Bild 2*). An den meisten Fernsehgeräten findet man das Video-Ausgangssignal an der AV- oder Scart-Buchse.

Videoausgang an der AV-Buchse:
– Signal: Stift 2
– Masse: Stift 3

Videoausgang an der Scart-Buchse:
– Signal: Stift 19
– Masse: Stift 17

Das Interface separiert aus dem Videosignal die übertragenen Dateninformationen und stellt diese an einer seriellen Schnittstelle (DB 25-Buchse) zur Verfügung. Genauso wie in der Datenfernübertragung (DFÜ) mit Akustik-Kopplern werden diese seriellen Informationen mit 300 Baud, 1 Startbit, 8 Datenbits und 2 Stopbits an den Rechner des Zuschauers weitergeleitet.
Der Computer muß dementsprechend über eine serielle Schnittstelle (RS-232C) und entsprechende DFÜ- oder Terminal-Software verfügen.
An dieser Stelle wollen wir Ihnen die für den Commodore C-64 bzw. -128 erforderlichen Hilfsmittel als speziell für Videodat zugeschnittenes Paket in Hard- und Software anbieten.



	Zeile im 1.Halbbild	Zeile im 2.Halbbild
	23	336
	24	337
Startbit	25	338
1. Datenbit	26	339
2. "	27	340
3. "	28	341
4. "	29	342
5. "	30	343
6. "	31	344
7. "	32	345
8. "	33	346
1. Stopbit	34	347
2. Stopbit	35	348
	36	349

Bild 1. So wird Videodat im Fernsehbildsignal übertragen

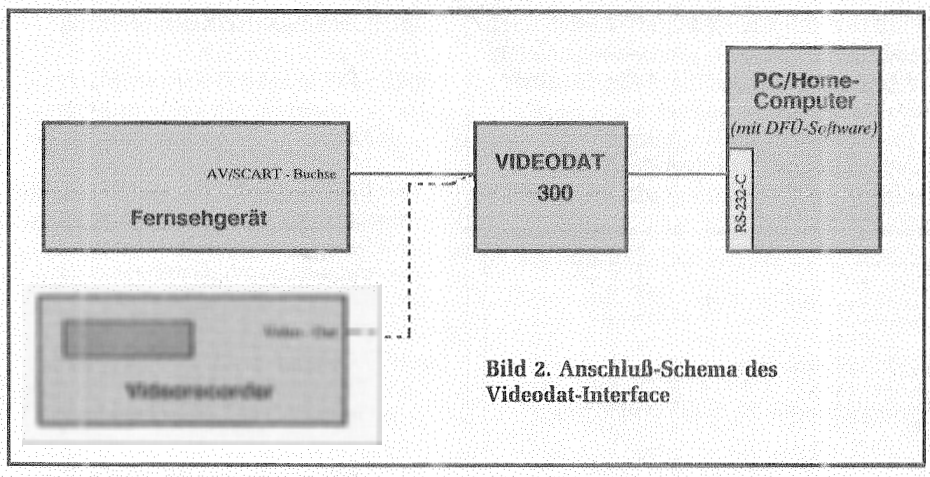


Bild 2. Anschluß-Schema des Videodat-Interface

Die Hardware

Die Commodore-Rechner C-64 und C-128 verfügen zwar jeweils über eine serielle Schnittstelle, es fehlt jedoch die Pegelanpassung an die RS232-Norm. Das ist für den Empfang bzw. die Auswertung der Videodat-Informationen unbedingt erforderlich. Dieses Problem läßt sich aber mit einer kleinen Schaltung lösen (siehe *Bild 3*).
Man kann sie direkt an der Federleiste für das User-Port anbauen. Die Schaltung wandelt den Pegelhub der RS-232C-Schnittstelle (±12 V) in einen TTL-Pegelhub (0 V/+5 V) um.

Die Software

Die Software besteht aus drei Basic-Programmen, die den einzelnen Arbeitsschritten angepaßt sind:

– das Empfangsprogramm
– der Editor
– der Konverter

Durch diese Gliederung wird der Arbeitsspeicher optimal für das Ablegen von Empfangsdaten genutzt.

Das Empfangsprogramm

Es hat die Aufgabe, die Daten auf der Diskette abzuspeichern, die über die serielle Schnittstelle eintreffen. Dabei gilt es, drei Hindernisse zu überwinden:

– das Basic-Geschwindigkeitsproblem
– die Korrektur des C-64-ASCII-Zeichensatzes
– die Kollisionen bei gleichzeitigem Betrieb von Diskette und RS-232C-Schnittstelle

Da der C-64 nicht die RS-232C-Schnittstelle gleichzeitig mit der Floppy bedienen kann, müssen die Empfangsdaten zwischengespeichert werden. Das Empfangsprogramm öffnet zunächst den Kanal für die RS-232C-Schnittstelle, danach den Fehlerkanal für die Floppy und begrenzt dann das untere Speicherende für das Ablegen von Strings (Empfangsdaten) auf 4000. Dadurch steht ein RAM-Bereich von 36 KByte zur Verfügung.
Durch die Anweisung „GET#2,A$" in Zeile 2030 wird die Schnittstelle abgefragt und gleichzeitig werden die Daten in den Stringspeicher gebracht.
Jedesmal, wenn der Basic-Interpreter eine Textvariable mit einem Zeichen lädt, legt er dieses automatisch in seinem String-Speicher ab. Das erfolgt vom oberen String-Speicherplatz abwärts. Diese Tatsache machen wir uns für die Ablage der empfangenen Dateninformationen zu nutze.
In Zeile 2020 wird die aktuelle Obergrenze abgefragt. Dadurch gelingt es, kontinuierlich mit 300 Baud Zeichen zu empfangen und abzuspeichern – bis zu einem Umfang von maximal 36 KByte, und das trotz der langsamen Arbeit von Basic.

Nach Ende der Videodat-Übertragung, daran erkennbar, daß mindestens 30 s keine Zeichen mehr empfangen worden sind, muß die Leertaste betätigt werden. Die im String-Speicher abgelegten Zeichen werden, nach Eingabe des File-Namens, mit PEEK gelesen. Danach werden sie in das C-64-ASCII-Format gewandelt und unter dem gewählten File-Namen als sequentielle Datei auf Diskette gespeichert. Das Unterprogramm in den Zeilen ab 20000 überprüft zuvor die Eingabe des File-Namens mit Hilfe des Floppy-Fehlerkanals. Es gibt im Fehlerfall eine Meldung auf dem Bildschirm aus.
Nach Ausgabe der Fehlermeldung veranlaßt der erzeugte RETURN-Code (Variable RC) das aufrufende Programm zur erneuten Abfrage des File-Namens – jedoch ohne die im String-Speicher befindlichen Empfangsdaten zu zerstören!
Das Programmlisting finden Sie in *Bild 4*.

Der Editor

Mit dem Editor können Sie die empfangenen Daten weiter bearbeiten und anpassen.
Die häufigste Anwendung wird das Zerlegen der gesamten Videodat-Datei in inhaltsgebundene Dateien sein. Über Videodat werden neben Textinformationen auch Computerprogramme im ASCII-Format gesendet. Diese müssen für die weitere Bearbeitung in eine eigene Datei gebracht werden.
Der Editor ist zeilenorientiert. Er besitzt zwei Betriebsebenen:

– die Kommando-Ebene „C" und
– die Eingabe-Ebene „I".

Nach RUN meldet sich der Editor mit seinem Kurzanleitungs-Menü „help" und verzweigt in den Kommando-Mo-

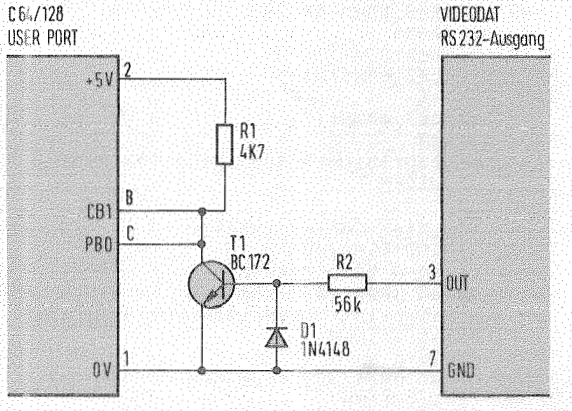


Bild 3. Hardware für den Anschluß des Videodat-Interface

dus „C...≫?“ („...“ steht für die aktuelle Schreibzeile). Im Kommando-Modus kann eine der auf dem Bildschirm angezeigten Funktionen durch Eingabe des Namens – z. B. load – aktiviert werden. Die Parameter, die für die gewählte Funktion erforderlich sind, werden nach Aufruf der Funktion programmgesteuert abgefragt. Bei „load“ wird die Eingabe des Dateinamens gefordert.

Der Editor besitzt folgende Funktionen:

```
1000 rem videodat-empfaenger
1010 rem
1020 rem (c) heike & manfred fillinger , den 5.9.86
1030 rem
1040 rem ini.
1050 open 2,2,0,chr$(6)+chr$(0):get#2,a$:rem rs232
1055 open15,8,15,"i0":rem disk-fehler-kanal
1060 poke53280,0:poke53281,0:printchr$(5):rem bildschirmfarben
1070 printchr$(14):rem kleinschrift
1080 printchr$(147);
1090 rem
1100 rem konstanten
1120 ed =  4000 : rem ram-pufferende
1130 rem
1500 rem display
1510 print:printchr$(30);:rem gruen
1520 print"  V I D E O D A T - E M P F A E N G E R"
1530 print"  ================================="
1540 print:printchr$(158);:rem gelb
1550 print"  Dieses Programm empfaengt von der "
1560 print"  RS232-Schnittstelle die VIDEODAT-"
1570 print"  Informationen und legt diese im"
1580 print"  Arbeitsspeicher (max.36K) ab.Die em-"
1590 print"  pfangenen Daten werden zeichenweise"
1600 print"  am unteren Bildrand angezeigt."
1610 print"  Wenn die VIDEODAT-Uebertragung be-"
1620 print"  endet ist, es wurden ca. 30 sec lang"
1630 print"  keine Zeichen mehr empfangen, be-"
1640 print"  taetigen Sie bitte die Leertaste."
1650 print"  Nach Eingabe eines Filenamens werden"
1660 print"  die im Arbeitsspeicher stehenden"
1670 print"  Daten auf Diskette gespeichert."
1680 print
2000 rem rs232 lesen
2020 an=peek(51)+256*peek(52)-1:rem obergrenze der strings
2025 i=an
2030 get#2,a$:if a$="" then 2090
2040 d%=asc(a$)
2050 poke 2023,d% and 95
2070 i=i-1
2080 if i<ed then print"Speicherueberlauf":goto mx=i+1:goto 3000
2090 get b$:if b$=" " then mx=i+1:goto 3000
2100 goto 2030
3000 rem save auf disk
3005 printchr$(147);
3010 printchr$(159);:rem blau
3020 print"Bitte Dateiname eingeben :";:input dn$
3030 open3,8,3,"@:"+dn$+",s,w"
3035 gosub20000:ifrc=1thenrc=0:goto3010
3040 print#3,chr$(13)
3050 for i=an to mx step-1
3060 d%=peek(i)
3070 e%=0:if d%<91 and d%>64 then e%=128
3080 if d%>96 and d%<129 then d%=d%-32
3090 if d%=8 then d%=20
3100 d%=d%+e%
3110 printchr$(d%);
3120 print#3,chr$(d%);
3130 next i
3140 print#3,chr$(10)
3150 close3
3170 print:printchr$(158):print"FERTIG"
3175 close15
3180 end
20000 rem disk-fehler pruefen
20010 input#15,en,em$
20020 if en=0 then return
20025 print chr$(150):rem rot
20030 print:print" Disketten-Fehler":print
20040 print en;em$
20050 close 3
20055 print chr$(30) :rem gruen
20060 rc=1:return
```

Bild 4.
Der Videodat-Empfänger

help: Anzeige der Kurzanleitung

list: Der geladene oder eingegebene Text wird mit Zeilennummern auf dem Bildschirm angezeigt. Die Anzeige kann jederzeit durch das Betätigen der Leertaste angehalten bzw. fortgesetzt werden.
Parameter: von Zeilennummer, bis Zeilennummer

llist: Gleiche Funktion wie „list“, Ausgabe über Drucker. Die Leertaste bleibt ohne Funktion!

print: Der geladene oder eingegebene Text wird ohne Zeilennummern auf dem Bildschirm angezeigt. Auch diese Anzeige kann mit der Leertaste angehalten bzw. fortgesetzt werden.
Parameter: von Zeilennummer, bis Zeilennummer.

lprint: Gleiche Funktion wie „print“, die Ausgabe erfolgt jedoch über Drucker. Die Leertaste bleibt ohne Funktion!

delete: Der Aufruf dieser Funktion ermöglicht das Löschen einer oder mehrerer Textzeilen. Er ist für das Isolieren bestimmter Videodat-Inhalte wichtig.
Parameter: von Zeilennummer, bis Zeilennummer.

change: Kürzel „c“. Damit können Zeichen oder Zeichenketten innerhalb eines gewählten Textblocks gelöscht oder ausgetauscht werden.
Parameter: von Zeilennummer, bis Zeilennummer, Urtext, Neutext.

copy: Mit „copy“ können einzelne Textzeilen oder Zeilenblöcke an eine andere Stelle kopiert werden. Das Original bleibt erhalten.
Parameter: von Zeilennummer, bis Zeilennummer, nach Zeilennummer.

move: Wie „copy“. Der ursprüngliche Text wird dabei gelöscht.
Parameter: von Zeilennummer, bis Zeilennummer, nach Zeilennummer.

dir: „dir“ zeigt das Inhaltsverzeichnis der Diskette mit Angabe der jeweils belegten Blöcke und der

freien Blöcke. „dir" beeinflußt den im Arbeitsspeicher befindlichen Text nicht. Mit der Leertaste kann die Ausgabe angehalten bzw. fortgesetzt werden.

save: Mit „save" speichert man den aktuellen Inhalt des Textspeichers als sequentielle Datei auf Diskette ab. Wenn der gleiche Dateiname schon im Inhaltsverzeichnis der Diskette aufgefunden wird, so wird durch diesen Funktionsaufruf die vorherige Datei überschrieben! Eventuell auftretende Diskettenfehler werden angezeigt. Nach dieser Anzeige wird, selbstverständlich ohne Zerstörung des aktuellen Textinhaltes, wieder zur Kommando-Ebene verzweigt.
Parameter: Dateiname.

load: Mit „load" wird von der Diskette eine sequentielle Datei, beginnend mit der ersten freien Textzeilennummer, eingelesen. Das ermöglicht das Zusammensetzen eines Textes aus Standard-Dateien. Diese Funktion ist für das Verbinden einer Basicode-2-Standard-Unterprogrammbibliothek mit einem Basicode-2-Programm interessant. Die Floppy-Fehlerbehandlung geschieht in der gleichen Weise, wie unter „save".
Parameter: Dateiname.

i...: Diese Eingabe bewirkt das Umschalten in die Eingabe-Ebene (Texterstellung). Der neu einzugebende Text wird ab Zeilennummer „..." an- bzw. eingefügt. Die Eingabe-Ebene kann durch „RETURN" ohne vorherige Eingabe eines Zeichens abgeschlossen werden. Einfacher Wechsel zwischen C/I-Ebene durch „RETURN".

exit: Kürzel „ex", beendet den Editor.

Da der Editor in Basic geschrieben ist, beanspruchen einige Funktionen eine lange Zeitspanne. Besonders zeitintensiv ist die „load"-Fuktion: Jedes Zeichen wird einzeln von der Diskette geladen, untersucht und im Textspeicher abgelegt. Wie bereits in der Beschreibung zum Videodat-Empfänger erwähnt, legt der Basic-Interpreter mit jeder Textvariablenzuweisung diesen Text im Stringspace ab. Dadurch wird dieser Speicherbereich sehr schnell mit unnütz abgelegten Zeichen gefüllt. Wenn dies der Fall ist, stoppt der Interpreter für mehrere Sekunden und bereinigt seinen Stringspace. Besonders unangenehm macht sich diese Eigenschaft beim Laden umfangreicher Texte bemerkbar. Das Laden einer 20 KByte langen Text-Datei dauert über zehn Minuten. Abhilfe könnte hier nur ein Basic-Compiler schaffen. Nach dem Compilieren des Editors kann dieser 10- bis 20mal schneller arbeiten. Das Listing ist in *Bild 5* enthalten.
Wenn man über Videodat ein Basicode-2-Programm [2] empfangen und als sepa-

```
1000 rem ***"vd-edit"***
1010 rem ein editor fuer videodat
1020 rem (c) heike und manfred fillinger , 8.9.1986
1030 open  15,8,15,"i0":rem  disk-fehler-kanal
1040 rem konstanten
1050 gz=1500:rem maximale textzeilenzahl
1060 zm=1:rem kleinste zeilenzahl
1065 r=6:rem linker papierrand bei lprint
1070 ta=8:rem tab-weite
1072 bs$=chr$(20):rem back-space(inst del)
1074 tb$=chr$(29):rem tabulator
1076 wr$=chr$(13):rem wagenruecklauf
1094 dim t$(1500)
1095 poke 53280,0:poke 53281,0:print chr$(30):rem bildschiermfarben
1096 print chr$(14):rem kleinschrift
1320 print chr$(147);
1330 print "            *** VD-EDIT ***"
1335 print
1999 goto 7000
2000 rem control-modul
2015 poke 54296,15:poke 54278,255:poke 54273,95:poke 54276,17
2016 for oo=1 to 200:next oo
2017 poke 54276,0:poke 54277,0:poke 54278,0:rem ton
2018 print
2020 print "c";zm;tab(5)">>";
2030 c$="":input c$
2040 if c$="" then n1=zm:goto 3040
2050 if left$(c$,1)="i" then :z=n1:goto 3000
2060 if c$="list" then gosub 15050:goto 4000
2070 if c$="llist" then gosub 15050:goto 4070
2075 sj=1
2080 if c$="print" then gosub 15050:goto 4120
2090 if c$="lprint" then gosub 15050:goto 4170
2100 if c$="delete" then gosub 15050:goto 5000
2110 if c$="save" then goto 6000
2120 if c$="load" then goto 6300
2130 if  c$="dir"  then  goto  14000
2140 if c$="help" then goto 7000
2150 if c$="change" or c$="c" then gosub 15110:goto 8000
2170 if c$="copy" then goto 10000
2180 if c$="move" then goto 11000
2190 if left$(c$,2)="ex" then close 15:end
2200 print "unzulaessiges Kommando !"
2220 goto 2000
3000 rem eingabe-modul
3020 gosub 15010
3040 z=n1
3050 gosub 15250
3060 print "i";z;tab(5)">>";
3070 gosub 15400
3080 if c$="" then zm=z:gosub 15330:goto 2000
3090 t$(z)=c$:z=z+1
3100 if z=zv then print "Textspeicher voll !"
3110 goto 3060
4000 rem ausgabe-modul
4020 rem list
4030 for i=n1 to n2
4040 print i;tab(4)":";t$(i)
4045 gosub  16000
4050 next i
4060 goto 2000
4070 rem llist
4075 open 4,4,7
4080 for i=n1 to n2
4090 print# 4,i;tab(4)":";t$(i)
4100 next i
4105 close 4
4110 goto 2000
4120 rem print
4130 for i=n1 to n2
```

Bild 5. Der Editor

rate Datei auf Diskette abgelegt hat, muß diese mit der Unterprogrammbibliothek von Basicode-2 verbunden werden. Dazu wird der Textspeicher zunächst mit „delete“ gelöscht. Dann wird zuerst die Unterprogrammbibliothek mit „load“ geladen und anschließend das zuvor separierte Basicode-2-Programm hinzugeladen. Dieses so zusammengesetzte ASCII-Programm-File wird nun mit „save“ auf der Diskette gespeichert.
Die Basicode-2-Unterprogrammbibliothek finden Sie in *Bild 6*.

Der Konverter

Wenn man über Videodat ein Programm für den C-64 oder ein Basicode-2-Programm empfangen und mit dem Editor als separate Datei abgespeichert hat, ist dieses noch nicht sofort als Basic-Programm lauffähig, da es sich ja um eine sequentielle Datei handelt. Eine solche ASCII-Programm-Datei kann mit dem Konverter in ein lauffähiges Programm umgewandelt werden.
Nach dem Programmstart wird die Ein-

```
 4140 print t$(i)
 4145 gosub 16000
 4150 next i
 4160 goto 2000
 4170 rem lprint
 4175 open 4,4,7
 4180 for i=n1 to n2
 4190 print# 4,tab(r)t$(i)
 4200 next i
 4205 close 4
 4210 goto 2000
 5000 rem delete-modul
 5020 if n2=nm then zm=zm-1:goto 2000
 5030 zv=n1
 5040 for i=n2+1 to zm
 5050 t$(zv)=t$(i)
 5060 zv=zv+1
 5070 next i
 5080 zm=zm-(n2-n1+1):goto 2000
 6000 rem disk.-modul
 6020 rem save
 6040 print "dateiname :";:input dn$
 6050 open 3,8,3,"@:"+dn$+",s,w"
 6053 gosub 20000:if rc=1 then rc=0:goto 2000
 6055 print# 3,wr$
 6080 for i=1 to zm-1
 6090 print# 3,t$(i)
 6100 next i
 6105 print# 3,chr$(10)
 6110 close 3
 6120 goto 2000
 6300 rem load
 6310 print "dateiname :";:input dn$
 6320 open 3,8,3,dn$+",s,r"
 6325 gosub 20000:if rc=1 then rc=0:goto 2000
 6330 i=zm:t$(i)=""
 6333 get #3,a$
 6335 if a$=wr$ then 6333
 6337 goto 6350
 6340 get #3,a$
 6350 if a$=wr$ then print t$(i):i=i+1:t$(i)="":goto 6370
 6360 t$(i)=t$(i)+a$
 6370 if st<>64 then 6340
 6380 close 3
 6385 if t$(i-1)=chr$(10) then t$(i-1)="":i=i-1
 6390 zm=i
 6400 goto 2000
 7000 rem kurzanleitungs-modul'help'
 7010 print "help   : zeigt diese Kurzanleitung"
 7030 print "list   : schreibt Text mit Z-nr. *)"
 7040 print "llist  : druckt Text mit Zeilennr."
 7050 print "print  : schreibt Text ohne Z-nr. *)"
 7060 print "lprint : druckt Text ohne Zeilennr."
 7070 print "delete : loescht Textzeilen"
 7080 print "(c)hange: Austausch von Zeichen bzw ZK."
 7100 print "copy   : kopiert Textzeilen im Text"
 7110 print "move   : verschiebt Textzeilen im Text"
 7120 print "save   : schreibt Text auf Diskette"
 7130 print "load   : liest Text von der Diskette"
 7140 print "dir    : Disketten-Inhalt *)"
 7150 print "i...   : Eingabe-Modus ab Textzeile..."
 7160 print "return  : Wechsel zwischen c-/i-modus"
 7165 print "(ex)it  : Ende"
 7170 print
 7174 print "*) Anhalten bzw. Weiter durch Leertaste"
 7176 print
 7180 goto 2000
 8000 rem change-modul
 8020 for i=n1 to n2
 8030 print :print "<";i;">";
 8040 j=1:c$="":p=1
 8050 gosub 8140
 8060 if j=0 then goto 8100
 8065 if j<>0 then print "*";
 8070 c$=c$+mid$(t$(i),p,j-p)+n2$
 8080 j=j+len(n1$):p=j
 8090 goto 8050
 8100 t$(i)=c$+mid$(t$(i),p,len(t$(i))-p+1)
 8110 next i
 8120 print
 8130 goto 2000
 8140 rem in$
 8150 for j=j to len(t$(i))-len(n1$)+1
 8160 if n1$=mid$(t$(i),j,len(n1$)) then return
 8170 next j
 8180 j=0:return
10000 rem copy-modul
10020 gosub 15170
10030 gosub 15250:j=n3
10040 for i=n1 to n1+n2-n3+1
10050 t$(i)=t$(j)
10060 j=j+1
10070 next i
10080 zm=i-1
10090 gosub 15330
10100 goto 2000
11000 rem move-modul
11020 gosub 15170
11030 gosub 15250:j=n3
11040 for i=n1 to n1+n2-n3+1
11050 t$(i)=t$(j)
11060 j=j+1
11070 next i
11080 zm=i-1
11090 gosub 15330
11100 if n1<n2 then n2=n2+zm-gz:n3=n3+zm-gz
11110 n1=n3
11120 zv=n1
11130 for i=n2+1 to zm
11140 t$(zv)=t$(i)
11150 zv=zv+1
11160 next i
11170 zm=zm-(n2-n1+1)
11180 goto 2000
14000 rem  dir
14010 open 1,8,0,"$0"
14020 gosub 20000:if rc=1 then rc=0:goto 2000
14030 print chr$(158):rem  gelb
14040 print "Anhalten/Weiter mit Leertaste":print
14050 get #1,a$,b$
14060 get #1,a$,b$
14070 get #1,a$,b$
14080 c=0
```

```
14090 gosub 16000
14100 if  a$<>"" then  c=asc(a$)
14110 if  b$<>"" then  c=c+asc(b$)*256
14120 print  mid$(str$(c),2);tab(3);
14130 get #1,b$:if  st<>0 then  14210
14140 if  b$<>chr$(34) then 14130
14150 get #1,b$:if  b$<>chr$(34) then  print  b$;:goto 14150
14160 get #1,b$:if  b$=chr$(32)  then  14160
14170 print  tab(18);:c$=""
14180 c$=c$+b$:get #1,b$:if b$<>""  then  14180
14190 print  left$(c$,3)
14200 if  st=0  then  14060
14210 print " Blocks free"
14220 close  1
14230 print  chr$(30) :rem  gruen
14240 goto 2000
15000 rem zentrales unterprogramm-modul
15010 rem i-parameter
15020 n1=val(right$(c$,len(c$)-1))
15030 if n1>zm then n1=zm
15040 return
15050 rem 2-parameter
15060 print "von Zeile , bis Zeile";:input n1,n2
15070 if (n2>zm-1) or (n2<1) then n2=zm-1
15080 if n1<1 then n1=1
15090 if n1>n2 then n1=n2
15100 return
15110 rem 2+2-parameter
15120 gosub 15050
15130 print "Urtext : ";:gosub 15400:n1$=c$
15140 print
15150 print "Neutext: ";:gosub 15400:n2$=c$
15160 return
15170 rem 3parameter
15180 gosub 15050
15190 n3=n1
15200 input "nach Zeile";n1
15210 if (n1>n3) and (n1<=2) then 15240
15220 if n1<=n3 then n2=gz-(zm-n2):n3=gz-(zm-n3)
15230 return
15240 print "Parameterfehler":goto 2000
15250 rem luecke aufbauen
15260 zv=gz
15270 for i=zm to n1 step -1
15280 t$(zv)=t$(i)
15290 zv=zv-1
15300 next i
15310 zv=zv+1
15320 return
15330 rem anfuegen
15340 for i=zv to gz
15350 t$(zm)=t$(i)
15360 zm=zm+1
15370 next i
15380 zm=zm-1
15390 return
15400 rem tastatur abfragen
15410 c$=""
15415 rem cursor unter get blinken
15420 poke 204,0:poke 198,0:wait 198,1:get in$
15422 poke 207,0:poke 204,1
15425 ky$=in$
15430 if (len(c$)<=0) and (ky$=bs$) then goto 15420
15440 if ky$=bs$ then gosub 15500
15450 if ky$=tb$ then gosub 15520
15460 if ky$>=" " then c$=c$+ky$:print ky$;:goto 15420
15470 if ky$<>wr$ then goto 15420
15480 print ky$;
15490 return
15500 if len(c$)>1 then c$=left$(c$,len(c$)-1):goto 15560
15510 c$="":print ky$;:return
15520 rem tab
15530 if (int(len(c$)/ta)=len(c$)/ta) and (ky$<>tb$) then return
15540 c$=c$+" ":print " ";:ky$=""
15550 goto 15530
15560 print ky$;:return
16000 rem  anhalten
16010 get bl$:if bl$=" " then gosub 16950
16020 return
16950 rem  warten auf leertaste
16960 get bl$:if bl$<>" " then 16960
16970 return
20000 rem  disk-fehler pruefen
20010 input# 15,en,em$
20020 if en=0 then return
20025 print chr$(150):rem  rot
20030 print :print " Disketten-Fehler":print
20040 print en;em$
20050 close 3
20055 print chr$(30):rem  gruen
20060 rc=1:return
```

```
10 goto1000
20 goto1010
100 print chr$(147);:return
110 poke211,ho:poke214,ve:sys58640:return
120 ho=peek(209)+256*peek(210)-1024+peek(211)
121 ve=int(ho/40):ho=ho-40*ve:return
200 getin$:return
210 getin$:ifin$=""goto210
220 return
250 poke54296,15:poke54278,255:poke54273,95:poke54276,17
251 foroo=1to200:nextoo
252 poke54276,0:poke54277,0:poke54278,0:return
260 rv=rnd(1):return
270 fr=fre(0):return
300 sr$=str$(sr):ifleft$(sr$,1)=" "thensr$=mid$(sr$,2)
301 return
310 sr$="":ifsr>=10^(ct-cn-sgn(cn))then318
311 ifsr<-.5*10^-cnthensr$="-"
312 sr$=sr$+mid$(str$(int(abs(sr)+.5*10^-cn)),2)
313 iflen(sr$)<ct-cn+(cn<>0)thensr$=" "+sr$:goto313
314 ifcn=0orabs(sr)>=1e10goto316
315 sr$=sr$+mid$(str$(1+abs(sr)-int(abs(sr))+.5*10^-cn),3,cn+1)
316 iflen(sr$)<ctthensr$=sr$+" ":goto316
317 iflen(sr$)>ctthensr$=""
318 iflen(sr$)<ctthensr$=sr$+"*":goto318
319 return
350 open4,4:print#4,sr$;:close4:return
360 open4,4:print#4:close4:return
```

Bild 6. Basicode-2-Unterprogrammbibliothek für den C-64

gabe des File-Namens der zu konvertierenden Datei erwartet. Häufig kommt es vor, daß Basic-Schlüsselwörter in Groß-Schrift empfangen wurden, obgleich im Rechner der Kleinschrift-Modus aktiviert war. Daher werden diese im ersten Arbeitsgang in Kleinschrift gewandelt und die gesamte ASCII-Programm-Datei wird wieder unter dem gleichen File-Namen abgespeichert.

Im zweiten Arbeitsschritt wird ein kleines Maschinenprogramm geladen. Der zugehörige Aufrufbefehl – SYS49155... – wird auf dem Bildschirm vorgegeben und anschließend der Basic-Programmspeicher gelöscht. Nach Betätigen der „RETURN"-Taste wird das Maschinenprogramm gestartet.

Das Maschinenprogramm liest die gewählte Datei ein und wandelt diese in ein lauffähiges Programm um. Am Ende dieses Schrittes erscheint auf dem Bildschirm die Meldung „ready". Im Arbeitsspeicher befindet sich nun ein lauffähiges Basic-Programm. Dieses kann ganz normal auf Diskette gespeichert oder mit „RUN" gestartet werden. *Bild 7* zeigt das Listing des Konverters.

Noch ein Tip

Wenn Sie den umgekehrten Weg gehen möchten und ein C-64-Programm in eine ASCII-Programm-Datei umwandeln wollen, können Sie dies sowohl mit als auch ohne den Konverter tun. Laden Sie zuerst das Basic-Programm in den Arbeitsspeicher.

Geben Sie dann folgende Befehlssequenz ein:

```
OPEN 3,8,3,"Name,S,W":CMD 3:LIST:
PRINT#3:CLOSE 3
```

Gleiches leistet jedoch auch das zuvor beschriebene Konvertier-Programm unter:

```
SYS 49152,"Name",8
```

Viel Spaß mit Videodat 300.

Literatur

[1] *Rohde, Ulrich:* WDR-Computerclub innovativ. mc 1986, Heft 6, Seite 66...67.

[2] Basicode, Die kompatible Computersprache. Otto-Maier-Verlag. (Basicode ist eine Entwicklung von Radio Hilversum, Red. Hobbyscoop.)


```
100 rem konvert
110 rem wandelt ein seq-file in ein lauffaehiges basicprogramm
120 rem (c)heike und manfred fillinger,20.10.1986
130 open 15,8,15,"i0":rem disk-fehlerkanal
140 dim t$(1000)
150 zk=0:rem keine zeichenkette erkannt
160 printchr$(147)
170 print chr$(14):rem kleinschrift
180 poke53280,0:poke53281,0:printchr$(30):rem bildschirmfarben
200 rem load und ascii/c64-ascii wandlung
210 print"Welche Datei soll konvertiert werden"
220 input dn$
230 open 3,8,3,dn$+",s,r"
240 gosub 20000:if rc=1 then rc=0:goto 200
250 i=1:t$(i)=""
260 get#3,a$:if a$=chr$(13)then 260
270 goto 290
280 get#3,a$
283 if a$=chr$(5) then 290
285 if a$<chr$(13) or a$>chr$(218) then 280
290 if a$=chr$(13) then print t$(i):i=i+1:zk=0:t$(i)="":goto 340
300 if a$=chr$(34) and zk=0 then zk=1:goto 320
310 if a$=chr$(34) and zk=1 then zk=0
320 if a$>chr$(192) and zk=0 then a$=chr$(asc(a$)-128)
330 t$(i)=t$(i)+a$
340 if st<>64 then 280
350 close 3
360 if t$(i-1)=chr$(10)then t$(i-1)="":i=i-1
370 zm=i
400 rem save
410 open 3,8,3,"@:"+dn$+",s,w"
415 gosub 20000
420 if rc=1 then rc=0:print"Konvertierung wiederholen!":close15:end
430 print#3,chr$(13)
440 for i=1 to zm-1
450 print#3,t$(i)
460 next i
470 print#3,chr$(10)
480 close 3
600 rem seq/pro-wandlung
610 gosub 900:rem maschinenprogramm laden
620 print"Bitte nach 1 Sekunde RETURN druecken"
630 print
640 print"sys49155,"chr$(34)+dn$+chr$(34)+",8"
650 print chr$(145);
660 print chr$(145);
670 print chr$(145);
680 new
900 rem pro/seq-maschinenprogramm
905 rem (c) w. schellenberger 4.85
906 rem programm in seq-file - sys49152,"name",8
907 rem das programm muss im speicher stehen
908 rem seq-file in programm - sys49155,"name",8
999 rem das programm steht anschliessend im speicher
1000 for i = 49152 to 49344
1010 read x : poke i,x : s=s+x : next
1020 data  76,123,192, 32, 55,192, 32, 80,192, 32,102,192
1030 data 162,  2, 32,198,255, 32,207,255, 32,207,255, 32
1040 data 128,164,169, 13, 32,210,255,169, 82, 32,210,255
1050 data 169, 13, 32,210,255, 32,204,255,169,  2, 32,195
1060 data 255, 32, 83,228, 76,157,227, 32,253,174,162,255
1070 data 142,188,192,238,188,192, 32,115,  0,174,188,192
1080 data 157,189,192,201, 34,208,240, 96,169,  2,168,162
1090 data   8, 32,186,255,173,188,192,162,189,160,192, 32
1100 data 189,255, 32,192,255, 96,169,122,141, 44,  3,169
1110 data 192,141, 45,  3,169, 26,141,  0,  3,169,192,141
1120 data   1,  3, 96, 32, 55,192,174,188,192,169, 44,157
1130 data 189,192,232,169, 83,157,189,192,232,169, 44,157
1140 data 189,192,232,169, 87,157,189,192,232,169, 34,157
1150 data 189,192,142,188,192, 32, 80,192, 32,102,192,162
1160 data   2, 32,201,255,169, 13, 32,210,255, 32,142,166
1170 data 169,  0, 32,156,166, 76, 26,192,  0,  0,247,215
1180 data   0
1190 if s <> 26297 then print "fehler in datas !!" : end
1200 return
20000 rem disk-fehler pruefen
20010 input#15,en,em$
20020 if en=0 then return
20030 print chr$(150):rem rot
20040 print:print"Disketten-Fehler":print
20050 print en;em$
20060 close 3
20070 print chr$(30):rem gruen
20080 rc=1:return
```

Bild 7. Dieses Programm konvertiert ein Programm, das als Text-Datei vorliegt, in ein lauffähiges Programm

Ulrich Cebulla

Grafik mit EGA

Teil 3

Jetzt noch ein Programmbeispiel, daß zeigen will, wie die Register der IBM-EGA benutzt werden können. Das Programmbeispiel geht davon aus, daß die IBM-EGA-Karte mindestens mit 128 KByte ausgerüstet ist. Für die 64-KByte-EGA-Karte müssen einige Änderungen vorgenommen werden, auf die am Schluß eingegangen wird.

Das Programmbeispiel *(Bild 1)* besteht aus vielen nützlichen Prozeduren (zu sehen in *Bild 2*), die zunächst beschrieben seien.

Procedure „graphics(true)"

Dieses Unterprogramm schaltet abhängig von true/false den Grafikmodus $0F für den Monochrom-Bildschirm, oder den Grafikmodus $10 für den Enhanced Graphics Bildschirm ein oder aus.

Procedure „speicherebene(ebene)"

Mit diesem Unterprogramm kann (können) die Speicherebene(n) ausgewählt werden, in der das Bild erzeugt werden soll. Normalerweise sind immer alle 4 Bildspeicherebenen ($0F) selektiert, sodaß man alle 16 Farben (aus 64) zur Verfügung hat. Will man jedoch, wie in diesem Programmbeispiel gezeigt, vier zweifarbige Bilder erzeugen, so kann mit dieser Prozedur die Auswahl der Speicherebene getroffen werden. Mit dieser Methode ist es möglich, die Teilbilder in den einzelnen Speicherebenen unabhängig voneinander zu verarbeiten (z. B. moving windows).

Im Programmbeispiel wird folgende Bildaufteilung benutzt:

a) Der Bildrahmen wird in die Bildspeicherebene 0 geschrieben.

b) Die Funktion F1=sin(x) wird in die Bildspeicherebene 1 geschrieben.

c) Die Funktion F2=sin(3x+p) wird in die Bildspeicherebene 2 geschrieben.

d) Die Funktion Σ F1,F2 wird in die Bildspeicherebene 3 geschrieben.

e) Die X-Achse wird in alle 4 Speicherebenen geschrieben.

Procedure „pset(x,y,farbe,funktion)"

Dieses Unterprogramm gleicht dem weit verbreiteten Basic-Befehl PSET. Hier ist jedoch noch ein vierter Funktionsparameter vorhanden, der für grafische Anwendungen sehr brauchbar ist. Die in der angewählten x,y- Position bereits vorhandene Farbinformation kann wie folgt mit der neuen verbunden werden:
Funktion
00 Keine Modifikation (überschreiben)
01 UND-Funktion,
10 ODER-Funktion,
11 XOR-Funktion.

Procedure „invertbild"

Diese Prozedur erlaubt das Invertieren des gesamten sichtbaren Bildes. Will man mehr invertieren, so braucht man nur die Länge in der MOVE- Instruktion entsprechend zu vergrößern. Die MOVE Instruktion führt ein MOVE auf sich selbst aus, das heißt jedes Bildschirm-Byte wird gelesen, mit XOR verarbeitet und wieder zurückgeschrieben.

Procedure „farbpalette(farbe,palettereg)"

Diese Prozedur benutzt den INT$10, ist also eine ganz normale BIOS Funktion. Setzt man z. B. das Palette Register 0, so erhält man die gewählte Farbe als sogenannte Hintergrundfarbe. Wie bereits beim Attribute Controller beschrieben, werden die Palette Register durch den Inhalt der 4 Bildspeicherebenen binär (0..15) adressiert.

Procedure „bildebene(ebene)"

Normalerweise werden immer alle 4 Bildspeicherebenen zur Adressierung der Palette Register durchgelassen. Mit dieser Prozedur ist es jedoch möglich, die Bildspeicherebenen zu maskieren. Will man z.B. nur die Bildspeicherebene 2 und 3 sehen, so muß $0C in den Parameter „ebene" gestellt werden. Damit kann auf einfache Weise der Bildschirm auch dunkel (Palette Register 0) gesteuert werden (ebene = 0), ohne den Bildspeicherinhalt zu verändern. (Bemerkung: Hier muß darauf geachtet werden, daß die Instruktion a := port[$3ba]; für Monochrom Modus a := port[$3da]; für Color Modus zu verwenden ist.)

Procedure „scroll(i)"

Dieses Unterprogramm erlaubt das vertikale Verschieben der Bildschirmzeilen nach oben oder unten (Scrollen). Mit dem Parameter „i" kann die Geschwindigkeit bestimmt werden. Diese Prozedur wird erst interessant, wenn man den logischen Bildschirmspeicher vergrößert, das heißt größer macht als den physikalischen Bildschirm. Auf dem physikalischen Bildschirm werden 80 Bytes × 8 Bit = 640 Punkte horizontal und 350

```
Program EGAgraf1;
{$I graphics.ega}                { Include file der Hilfsprogramme }
{$I prtscr.ega}                  { Bildschirm drucken mit ITOH 8510 SP }

const pi = 3.14142;
var
   a1,a2:integer;                { Amplitude }
   f1:integer;                   { Frequenz 1 }
   f2:integer;                   { Frequenz 2 }
   x,y,y1,y2,y3:integer;         { x/y Koordinaten }
   phase:real;                   { Phasenverschiebung }
   z:integer;                    { Lage der x-Achse }
   vert:integer;                 { vertikale Bildhöhe }

   farb1:byte;                   { Farbe der Punkte }
   farb2:byte;
   farb3:byte;
   taste:char;
begin

   graphics(true);             { Grafikmodus einschalten }

                               { Zeilenzahl vert = :          }
```

Bild 1. Das Beispielprogramm, das viele Prozeduren benutzt

```
vert :=405;                    { 350 horizotale Zeilen/Bildschirm }
                               { 32736 : 80 = 409 Zeilen/Seite (64k EGA) }
                               { 65535 : 80 = 819 Zeilen max. (128k EGA) }
startzeile :=0;                { startzeile 0 = Adresse $A000:0          }

{===========Bildaufbau in verschiedenen Bildspeicherebenen====================}

Speicherebene(1);              { Speicherebene 0 setzen}

for x := 0 to 9
   do for y := 0 to vert
      do pset(x,y,3,0);        { 10 Zeilen vertikal links, Speicherebene 0}
for x := 630 to 639
   do for y := 0 to vert
      do pset(x,y,3,0);        { 10 Zeilen vertikal rechts, Speicherebene 0}
for x := 10 to 629
   do for y := 0 to 9
      do pset(x,y,3,0);        { 10 Zeilen horizontal oben, Speicherebene 0}
for x := 10 to 629
   do for y := vert-10 to vert
      do pset(x,y,3,0);        { 10 Zeilen horizontal unten, Speicherebene 0}

   z := vert div 2;            { Position der x-Achse }
{------------------Bildaufbau der x-Achse in allen 4 Speicherebenen----------}
   Speicherebene(15);          { Speicherebene 0..3 setzen }

   for x := 10 to 629
      do pset(x,z,15,0);       { x-Achse, Speicherebene 0..3 (Farbe weiss) }
{------------------------------------------------------------------------------}
   a1 := 120;                  { Amplitude 1 }
   a2 := 45;                   { Amplitude 2 }
   farb1 :=15;
   farb2 :=15;
   farb3 :=15;

   f1 := 1;                    { Frequenzfaktor * 1 }
   f2 := 3;                    { Frequenzfaktor * 3 }

   phase := pi/2;              { Phasenverschiebung von f2 }


gotoxy(30,24); write('Bildhöhe = ',vert+1,' Zeilen');
gotoxy(17,25); write('I B M  EGA  Bildschirmausdruck im Grafik Modus');

for x :=10 to 629
do begin

{-----------------------Bildaufbau nur in der Speicherebene 1 ----------------}
   Speicherebene(2);           { Speicherebene 1 setzen }

   y1 := trunc(z-(sin(2*pi/640*x*f1)*a1));
   pset(x,y1,farb1,0);         { F1 Sinus zeichnen, Speicherebene 1}

{-----------------------Bildaufbau nur in der Speicherebene 2 ----------------}
   Speicherebene(4);           { Speicherebene 2 setzen }

   y2 := trunc(z-(sin(2*pi/640*x*f2+phase)*a2));
   pset(x,y2,farb2,0);         { F2 Sinus zeichnen, Speicherebene 2}

{-----------------------Bildaufbau hur in der Speicherebene 3 ----------------}
   Speicherebene(8);           { Speicherebene 3 setzen }

   y3:= y1 + y2 - z;           { Σ F1 , F2          }
   pset(x,y3,farb3,0);         { Σ F1 , F2 zeichnen, Speicherebene 3}
   end;

{==============================================================================}

gotoxy(36,1); write('Sinus');
gotoxy(6,6); write('f1=sin(x)');
gotoxy(6,16); write('f2=sin(3x+p)');
gotoxy(31,5); write(chr(228),' f1,f2');
```

```
{-------------------Alle Seicherebene werden gleichzeitig angezeigt-----------}
repeat
     read(kbd,taste);          { Auswahl der Speicherebenen uber die  }
                               { Tasten 1..9.                         }
       case ord(taste) of
       48: Invertbild;         { Taste 0 Bildschirm invertieren       }
       49: Bildebene(1);       { Taste 1 Bildebene 0 anzeigen         }
       50: Bildebene(2);       { Taste 2 Bildebene 1 anzeigen         }
       51: Bildebene(4);       { Taste 3 Bildebene 2 anzeigen         }
       52: Bildebene(8);       { Taste 4 Bildebene 3 anzeigen         }
       53: Bildebene(3);       { Taste 5 Bildebene 0 und 1 anzeigen   }
       54: Bildebene(5);       { Taste 6 Bildebene 0 und 2 anzeigen   }
       55: Bildebene(6);       { Taste 7 Bildebene 1 und 2 anzeigen   }
       56: Bildebene(7);       { Taste 8 Bildebene 0,1 und 2 anzeigen }
       57: Bildebene(15);      { Taste 9 alle Bildebenen anzeigen     }
      112: Prtscr(3,vert+1);   { Taste p Bildschirm drucken  0 und 1  }
       72: Scroll(1);          { Taste cursor up Bildschirm scrollen  }
       80: Scroll(1);          { Taste cursor down Bildschirm scrollen}
       end;
until taste = 'E';             { Taste "E" Programmende }
   graphics(false);            { Grafikmodus ausschalten }

end.
```

```
{* Filename: graphics.ega *}
const
     egabase=$A000;                { Startadresse des Bildschirmspeichers }
var
   savemode,startzeile,i,j :integer;


{----------------------Graphics Modus EIN/AUS schalten-------------------------}
procedure graphics(on:boolean);

type regs=record case integer of
            1:(ax,bx,cx,dx,bp,si,di,ds,es,flgs: integer);
            2:(al,ah,bl,bh,cl,ch,dl,dh: byte);
          end;
  var reg:regs;
begin
    if on                        { Grafiksmode einschalten }
    then begin
         reg.ax:=$0F00;          { AH = OF lesen des EGA Status }
         intr($10,reg);
         savemode:=reg.al;       { EGA mode in AL }

         reg.ax:=$0010;          { EGA mode OF Monochrome Bildschirm }
                                 { !!!!!    10 Enhanced Color Display }
         intr($10,reg);          { EGA BIOS Intrrupt $10 }
         end
    else begin                   { Grafiks Mode ausschalten }
         reg.ah:=0;
         reg.ax:=savemode;
         intr($10,reg);          { EGA BIOS Interrupt $10 }
         end;
end;


{----------------------Pascal Punkt setzen-------------------------------------}
{ Punkt setzen an der Position x/y, Farbe = color, Funktion = funktion         }

procedure pset(x,y:integer;color,funktion:byte);
var offset   :integer;
    bitmask:byte;
```

Bild 2. Der Grundbestand an Routinen

```
begin

bitmask   := $80 shr (x and 7 );
offset    := (y * 80)+(x shr 3);

                              { GK = Grafik -Kontroler }

      port[$3ce] := 0;        { GK SET/RESET Register 0 }
      port[$3cf] := color;    { Farbe ins SET/REST Register }
      port[$3ce] := 1;        { GK ENABLE SET/RESET Register 1 }
      port[$3cf] := $0F;      { erlaube alle Planes }
      port[$3ce] := 3;        { GK Rotate Register 3}
      port[$3cf] := (funktion and $03) shl 3;
                              { Funktion  00 unmodifiziert          }
                              {           01 UND mit gelesenen Daten }
                              {           10 OR  "        "      "   }
                              {           11 XOR "        "      "   }

      port[$3ce] := 8;        { GK BIT MASK Reg. 8 }
      port[$3cf] := bitmask;  { PSET nur das durch x/y ausgewaehlte bit }

      mem[egabase:offset]:=mem[egabase:offset]; { read/write, siehe Text     }
                                                { ueber den Grafik Controller}

      port[$3cf] := $FF;      { erlaube wieder alle bits in GK Register 8 }

      port[$3ce] := 1;        { GK ENABLE SET/RESET Register 1 }
      port[$3cf] := 0;        { disable alle Planes }
      port[$3ce] := 3;        { GK Mode Register 3 }
      port[$3cf] := 0;        { Funktion 00        }
end;


{----------------------BIOS Punkt setzen-----------------------------------------}
procedure bset(x,y,color:integer); { Setze einen Punkt an die Stelle x/y       }
                                   { mit der Farbe = color (BIOS intr10 AH=0C) }
                                   { Diese Procedure ist 2.17 mal langsamer    }
                                   { als die Pascal "pset".                    }
                                   { Ist das bit 7 der color var 1, so wird    }
                                   { noch die Funktion XOR ausgefuehrt.        }
type regs=record case integer of
            1:(ax,bx,cx,dx,bp,si,di,ds,es,flgs: integer);
            2:(al,ah,bl,bh,cl,ch,dl,dh: byte);
          end;
  var reg:regs;
begin
     reg.ah:=$0C;          { Punkt setzen }
     reg.al:=color;        { Farbe        }
     reg.cx:=x;            { x-Position   }
     reg.dx:=y;            { y-Position   }
     intr($10,reg);        { Interrupt 10 }
end;

{----------------------Bildschirm invertiern-------------------------------------}

procedure invertbild;
  begin
    port[$3ce]:=0;                    { GK Register 0                    }
    port[$3cf]:=15;                   { alle bits set/reset              }
    port[$3ce]:=1;                    { GK Register 1                    }
    port[$3cf]:=15;                   { erlaube alle bits                }
    port[$3ce]:=3;                    { GK Register 3                    }
    port[$3cf]:=$18;                  { XOR Funktion                     }
                                      { invertiere alle Bytes des ganzen }
    move(mem[egabase:0],mem[egabase:0],28000);
                                      {       Bildschirmspeichers        }
    port[$3cf]:=0;                    { null Funktion                    }
    port[$3ce]:=1;                    { GK Register 1                    }
    port[$3cf]:=0;                    { disable set/reset                }

  end;
```

```
{----------------------Farbpalettenregister setzen-----------------------------}

procedure farbpalette(farbe,palettereg:integer);
  var regs:record case integer of
             1:(ax,bx,cx,dx,bp,si,di,ds,es,flgs: integer);
             2:(al,ah,bl,bh,cl,ch,dl,dh: byte);
           end;
begin
  with regs do
    begin                          { Farbpaletten Register setzen }
      ax:=$1000;
      bl:=palettereg and $0F;
      bh:=farbe      and $3F;
      intr($10,regs);
    end;
end;

{----------------------Schreibmaske für Speicherebene/n-------------------------}
{ Note: Nur in die hier selektierte/n Bildspeicherebene/n kann mit der "pset" }
{       geschrieben werden.                                                   }

procedure speicherebene(ebene:byte);
begin
     port[$3c4]:=2;       { Sequencer Adress Register= Map Mask Reg }
     port[$3c5]:=ebene;   { Map Mask Register, Schreibmaske für Speicherebene/n}
end;
{----------------------Bildebene/n-Auswahl für Bildschirm-----------------------}
{ Note: Nur die hier ausgewählte/n BildSpeicherebene/n werden auf dem Bild-   }
{       schirm angezeigt.                                                     }

procedure bildebene(ebene:byte);
var a:byte;
begin
     a := port[$3da];              { Adressmodus, Attribute Kontroller       }
                                   { Note: port[$3ba] = monochrome Modus     }
                                   {       port[$3da] = color Modus          }
     port[$3c0] := $12;            { Color Plane Enable Register, Display AUS}
     port[$3c0] := ebene,          { Display Speicherebene/n                 }
     port[$3c0] := $20;            { Display EIN                             }
End;

{--------------------Bildschirm nach oben/unten scrollen----------------------}
{ Note: Nur mit der Leertaste kann diese Procedure wieder verlassen werden   }

procedure scroll(i:byte);
var taste: char;

  regs:record case integer of
       1:(ax,bx,cx,dx,bp,di,ds,es,flgs:integer);
       2:(al,ah,bl,bh:byte);
       end;
begin
regs.ax := startzeile;
repeat
  read(kbd,taste);
  case ord(taste) of
     72 : regs.ax := regs.ax + 80*i;    { i * Zeilen aufwärts,cursor taste up}
     80 : regs.ax := regs.ax - 80*i;    { i * Zeilen abwärts,cursor taste down}
  end;
                                  { Note: port $3b4/5 = monochrome Modus }
                                  {       port $3d4/5 = color Modus      }
  port[$3d4] := $0c;              { CRT Kontroller Register $0C }
  port[$3d5] := regs.ah;          { Start Adresse high }
  port[$3d4] := $0d;              { CRT Kontroller Register $0D }
  port[$3d5] := regs.al;          { Start Adresse low  }

until taste=' ';                  { Warteschleife, bis die Leertaste gedrückt wird}
startzeile := regs.ax;            { Start Zeile speichern }
  end;
```

Scan Lines vertikal angezeigt. Das heißt, eine Bildschirmseite besteht aus 80 × 350 = 28000 abzubildenden Bytes. Für eine Bildschirmseite sind auf der Karte aber 32736 Bildschirm- Bytes reserviert, was 32736 : 80 = 409 Scan Lines entspricht. Im Programmbeispiel sind nun 406 (00..405) Scan Lines benutzt. Das Unterprogramm „scroll(i)" zeigt die restlichen Zeilen. Mit den beiden Cursor-Tasten up/down kann der sichtbare Bildschirm nach oben oder unten bewegt werden. Die Bilder zeigen einmal 350 und einmal 406 Zeilen (Scan lines). Bei einem 256-KByte-Bildschirmspeicher stehen zwei Bildschirmseiten, oder 65536 Bildschirm-Bytes, oder 65536 : 80 = 819 Scan Lines zur Verfügung. Es ist selbstverständlich auch möglich, den Bildschirmspeicher horizontal zu vergrößern (größer als 80 Bytes), doch das soll hier nicht weiter diskutiert werden.

Procedure „prtscr(ebene,vertikal)"

Diese Prozedur *(Bild 3)* erlaubt es, den logischen Bildschirm auf einem Matrixdrucker auszudrucken. Mit dem Parameter„ebene" kann ausgewählt werden, welche Speicherebene/n gedruckt werden sollen. Mit dem Parameter „vertikal" wird die vertikale Ausdehnung (normalerweise 349) des logischen Bildschirmes angegeben. Im Programmbeispiel wurde ein ITOH 8510 SP benutzt. Um die Prozedur an andere Drucker anzupassen ist nur die Initialisierung des Druckers im ersten Teil zu ändern. Alle Bilder des Programmbeispiels wurden mit dieser Prozedur erstellt. Der Parameter 'ebene' hat folgende Auflösung:

ebene
0000 nichts drucken
0001 Speicherebene 0 drucken,
0010 Speicherebene 1 drucken,
0100 Speicherebene 2 drucken,
1000 Speicherebene 3 drucken,
....
1111 alle Speicherebenen drucken.

Werden zum Beispiel alle Speicherebenen gedruckt, so wird ein weißer Bildpunkt viermal (alle vier Speicherebenen sind „1") gedruckt.

Grafik und Text

Wie in den Ausdrucken der Teilbilder *(Bild 4 bis 9)* zu sehen ist, erscheint der Text in den Speicherebenen 1, 2 und 3. Das liegt daran, daß der Text mit einer normalen Pascal-Anweisung Write/ln mit der Farbe gelb ausgeführt wurde. Das Turbo Pascal benutzt hier offen-

```
procedure prtscr(planes:byte;vertikal:integer);
var
   esc,cmd,n3,n2,n1,n0,data,cr,lf:char;
   hor,ver,hbit,bit,scan,shrcnt,i:integer;
   egabytes :  array[0..6] of      byte;
   prtbyte,prtbit,plane,mask      :byte;

begin
{-------------------------ITOH 8510 in Grafikmodus bringen---------------------}
esc:=chr(27);
cmd:=chr(99);          { software reset }
n0:=chr(49);
write(lst,esc,cmd,n0);
delay(2000);           { delay 2 sec }

cmd:=chr(62);          { unidir }
write(lst,esc,cmd);

cmd:=chr(90);          { nur CR (kein LF) }
n0:=chr(128);
n1:=chr(0);
write(lst,esc,cmd,n0,n1);

cmd:=chr(84);          { N/144 inch Line Feed Pitch }
n1:=chr(49);
n0:=chr(52);           { 7 * 2/144 inch }
write(lst,esc,cmd,n1,n0);

cmd:=chr(83);          { Bit image Graphics }
n3:=chr(48);           {        0           }
n2:=chr(54);           {        6           }
n1:=chr(52);           {        4           }
n0:=chr(48);           {        0           }

cr:=chr(13);           { CR }
lf:=chr(10);           { LF }
scan:=0;               { Bildschirmanfang   }

{-------------------------------------------------------------------------------}
{  Es werden die unteren 7 Nadeln (350 : 7 ='50 scans) zur gleichen Zeit uni-  }
{  directional gedruckt. Die vertikale Zeilenzahl muß daher immmer ein viel-   }
{  faches von 7 sein, z.B. vert = 405 (00...405)                               }
{-------------------------------------------------------------------------------}
planes := planes and $0F;              { nur 0..15 ist gültig }

for i:= 0 to ((vertikal div 7) - 1)
do begin
   mask := 1;
   plane := 0;

   while mask < 9                      { max 4 scans }
   do begin
      if planes and mask <> 0 then
      begin
         write(lst,esc,cmd,n3,n2,n1,n0); { Start Printer Grafik  Operation }

         port[$3ce]:=4;                { Grafik  Kontr. Reg. 4  }
         port[$3cf]:=plane;            { Speicherebene selekt.  }

         for hor:=0 to 79              { 80 Bytes horizontal    }
         do begin
            for ver:=0 to 6            { 7 Bytes vertikal       }
            do
              egabytes[ver]:=mem[$a000:scan+hor+ver*80];  {Speicherebene lesen}
            for hbit:=0 to 7           { 8 bits horizontal/Byte }
            do begin
               prtbyte:=0;
               shrcnt:=7;
               for bit:=0 to 6         { 7 bits vertical/Printer-scan }
               do begin
                  prtbit:=egabytes[bit] and $80;
                  prtbit:=prtbit shr shrcnt;
                  prtbyte:=prtbyte or prtbit;          { 7 bits assemblieren }
                  egabytes[bit]:=egabytes[bit] shl 1;
                  shrcnt:=shrcnt - 1;
                  end;
               write(lst,chr(prtbyte));                { drucken }
               end;
            end;
            write(lst,cr);  { CR nach 640 bytes (Ende einer 7bit vert.Bildzeile)
         end;
         mask := mask shl 1;
         plane := plane + 1;  { nächste Speicherebene }
      end;
      scan:=scan + 560;       { scan um 7 * 80 = 560 erhöhen }
      write(lst,lf);          { LF zur nächsten Printerzeile }
   end;

cmd:=chr(99);          { software reset des ITHO Printers }
n0:=chr(49);
write(lst,esc,cmd,n0);

port[$3ce]:=4;
port[$3cf]:=0;         { Read Mode 0 }
end;
```

Bild 3. Bildschirmebenen drucken

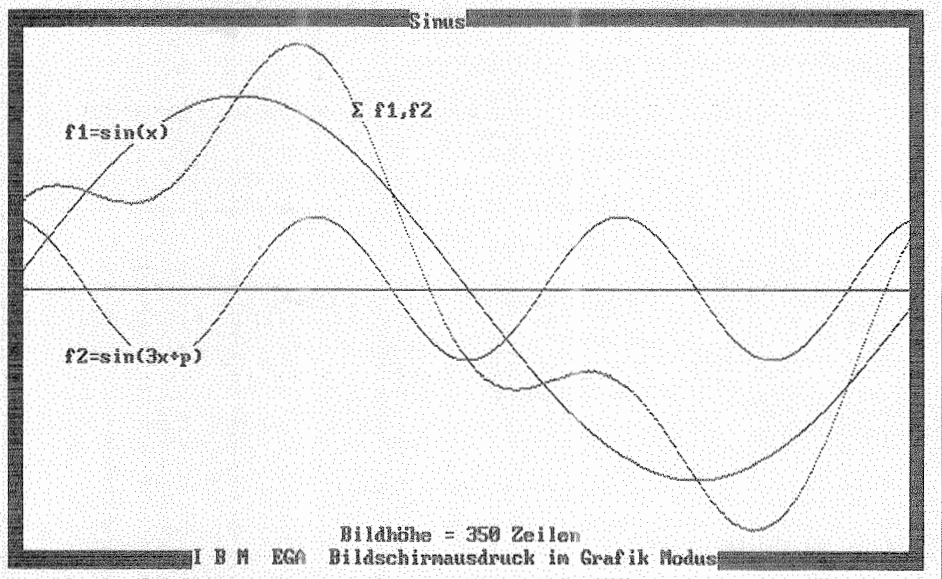


Bild 4. Schirm, vertikal mit 350 Zeilen gedruckt

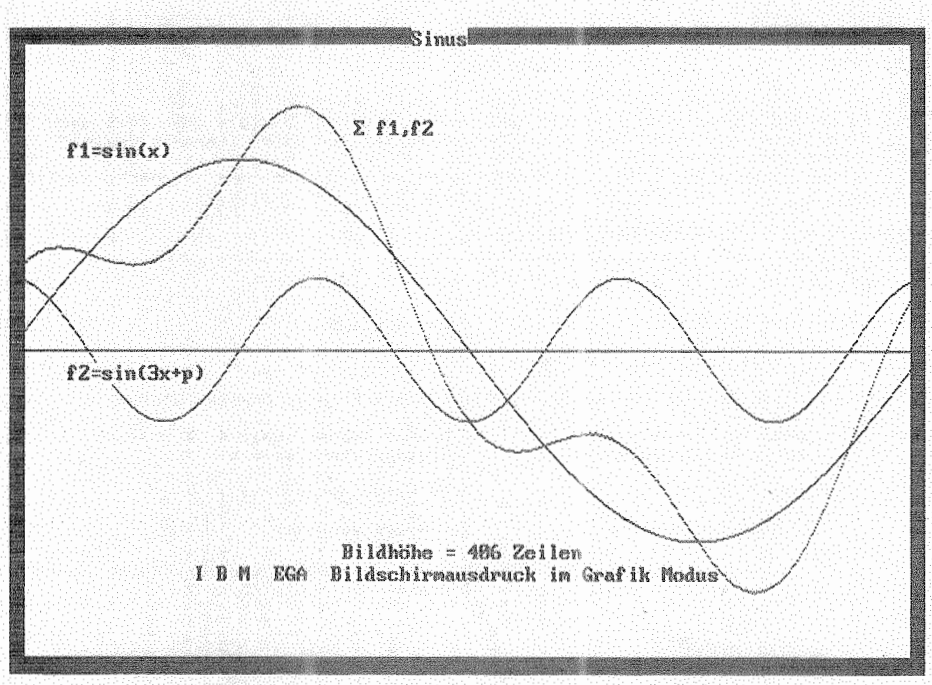


Bild 5. Schirm, aus vier Ebenen zusammengesetzt, 406 Scan Lines, Bildseite = 32 KByte

sichtlich die BIOS-Funktionen INT$10 AH=9/A (Write Character), die für die Farbe das Map Mask Register des Sequencers ($3c5.2) benutzt. Das ist das gleiche Register, das in der Prozedur „speicherebene" für die Selektion der Bildspeicherebene benutzt wird. Will man den Text bei grafischer Anwendung aber nur in einer bestimmten Bildspeicherebene haben, so muß man die Pascal Write/ln Instruktion auf eine eigene (kleine) Prozedur umleiten. Darauf, und auf einige andere interessante Details wird im nächsten Artikel näher eingegangen.

IBM-EGA mit 64 KByte Bildschirmspeicher

Das Odd/Even-Bit im Mode Register ($3cf.5) des Grafik-Controllers muß immer den Wert 1 haben (wie vom BIOS initialisiert).

Das Map Mask Register($3c5.2) des Sequencers ist wie folgt zu benutzen:

binär
0011 = Bildspeicherebene 0
1100 = Bildspeicherebene 2
1111 = Bildspeicherebene 0/2

Als Farbe sind nur gültig:
hex Monochorm/Color
0 = schwarz
3 = video/blau
C = blinking video/rot
F = intensified video/weiß

Bei Procedure „bildebene(ebene)" ist wie folgt zu verfahren.

binär
0001 = Bildebene 0 anzeigen
0100 = Bildebene 2 anzeigen
0101 = Bildebene 0/2 anzeigen 0101

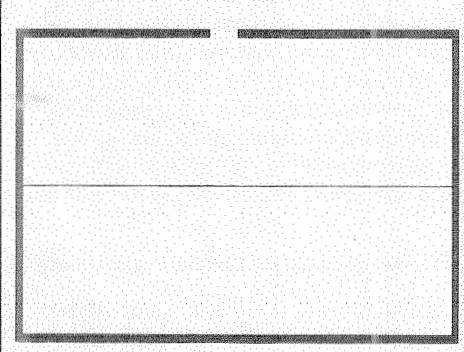


Bild 6. Ebene 0 aus Bild 5

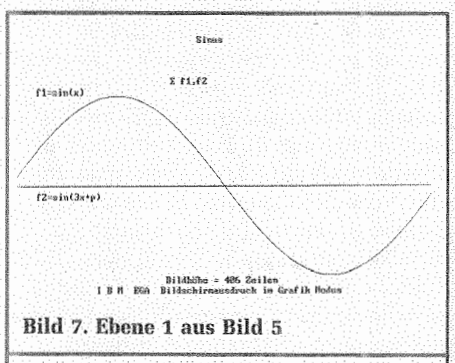


Bild 7. Ebene 1 aus Bild 5

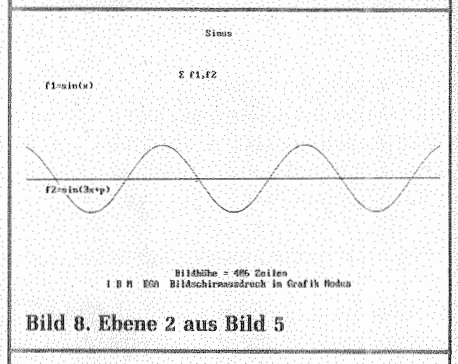


Bild 8. Ebene 2 aus Bild 5

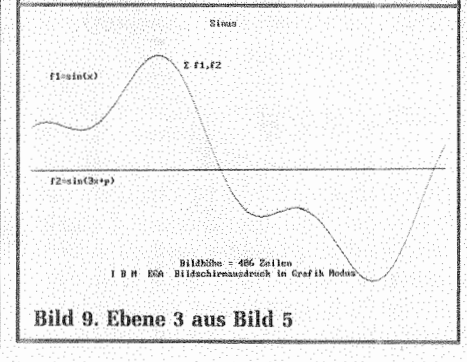


Bild 9. Ebene 3 aus Bild 5

Jürgen Schröder

Cursor ändern unter MS-DOS

Wie werden Parameter übergeben?

Mit einem DOS-Interrupt kann man den Cursor auf jede beliebige Größe einstellen, so daß der Cursor auch nach einer langen Nacht zu erkennen ist. Der Autor zeigt ausführlich, wie Parameter übergeben werden.

Mit dem MS-DOS-Interrupt 10H läßt sich leicht die Cursorgröße ändern. Register AH wird mit 01H geladen (Funktion 1). Es sind Anfangszeile (Übergabe durch das Register CH) und Endzeile (Register CL) bestimmbar. Durch die Anweisung CURSOR X, Y werden die Parameter in die genannten Register übernommen und der Interrupt 10H ausgelöst.

Die Interpretation der Parameter wurde variabel erstellt, damit sie auch für andere Zwecke übernommen werden kann. So sind die jeweiligen Maximalwerte (hier 13) programmierbar, eine Fehlerprüfung ist ausführlich im Programm kommentiert. Fehlerhafte Eingaben haben eine entsprechende Fehlermeldung zur Folge.

Wird CURSOR ohne Parameter eingegeben, erscheint der gewohnte Cursor.

Ein COM-Programm ist kürzer

Durch die ORG-Anweisung beginnt der Programmcode ab 100H. Ein Stack-Segment fehlt. So ist die Voraussetzung zur Erstellung eines COM-Programms geschaffen. Weitere Erläuterungen dazu sind in der mc-Serie „MS-DOS Intern" [1] zu entnehmen.

Zuerst wird überprüft, ob überhaupt Parameter vorhanden sind. Die Anzahl der dem Programm folgenden Parameter steht immer im Codesegment an der Stelle 80H. Ist der Wert 0H, so wird die Normaleinstellung vollzogen, andernfalls werden die folgenden Parameter untersucht und bei Richtigkeit in INT übernommen. In DOS wird das Programm durch den Interrupt 21H, Funktion 4CH beendet.

So wird CURSOR lauffähig

Das Assembler-Programm (*Bild 1*) kann mit EDLIN oder einem besseren Textverarbeitungsprogramm eingegeben werden. Mit einem Assembler wird ein Objektcode erzeugt, der wiederum mit dem Linker in eine EXE-Datei verwandelt wird. Der Linker wird ein fehlendes Stack-Segment beklagen, aber das können wir hier übersehen, da EXE2BIN die endgültige COM-Datei (*Bild 2*) erzeugt. Ein Beispiel für die Umwandlung von der Textdatei, z. B. CURSOR.ASM:

MASM CURSOR (4mal RETURN)
LINK CURSOR (4mal RETURN)
EXE2BIN CURSOR.EXE CURSOR.COM

Nun wollen wir unseren neuen MS-DOS-Befehl ausprobieren. CURSOR 0,13 erzeugt z. B. einen Blockcursor. Für dBase-III-Anwender sei bemerkt, daß natürlich mit RUNCURSOR X, Y der Cursor auch von dBase aus verändert werden kann.

Literatur

[1] *Smode, D.*: MS-DOS Intern. mc 1985, Heft 10.

```
code segment
                assume cs:code, ds: code
                org 100h                    ;weil .COM - Programm
start:          xor bx, bx
                mov di, 80h
                mov cl, byte ptr [di]       ;Parameter vorhanden?
                xor ch, ch
                cmp cl, 0
                jnz check_parameter
                jmp cursor_normal
check_parameter: inc di                     ;naechstes Zeichen
                mov al, [di]
                cmp al, ' '                 ;Leerzeichen?
                jz next_parameter           ;ja, naechstes Zeichen
                cmp al, ','                 ;Komma?
                jnz zahl_check              ;nein, Zahl interpretieren
                inc bh                      ;ja, zweiter Parameter (bh = 1)
                xor bl, bl                  ;folgt, Ziffernzahl = 0
                jmp next_parameter
zahl_checks:    cmp al, 48                  ;Ueberpruefung, ob Eingbe = Ziffer
                jl fehler_hinweis           ;<0? ja, Fehler
                cmp al, 57                  ;>9? ja, Fehler
                jg fehler_hinweis
                sub al, 48                  ;ASCII = Zahl
                cmp bh, 0                   ;erster Parameter (bh = 0)
                jnz parameter_2             ;nein, zweiter Parameter
parameter_1:    inc bl                      ;Ziffernanzahl inkrementieren
                cmp bl, 3                   ;bei 3 Ziffern:
                jz fehler_hinweis           ;Fehler
                cmp bl, 1                   ;1 Ziffer?
                jz para_1_1                 ;ja, abspeichern
                mov dl, al                  ;nein, 2. Ziffer in dl retten
                mov al, para1               ;1. Ziffer holen
                mov dh, 10                  ;mit 10
                mul dh                      ;multiplizieren
                add al, dl                  ;2. Ziffer addieren
para_1_1:       mov para1, al               ;1. Parameter abspeichern
                jmp next_parameter
parameter_2:    add bl, 1                   ;gleicher Vorgang wie
                cmp bl, 3                   ;parameter_1
                jz fehler_hinweis
                cmp bl, 1
                jz para_2_1
                mov dl, al
                mov al, para2
```

Bild 1. Der Quellcode von CURSOR

```
                    mov dh, 10
                    mul dh
                    add al, dl
para_2_1:           mov para2, al
next_parameter:     loop check_parameter        ;wenn dx <>0
fehler_check:       cmp bh, 0                   ;Komma vorhanden?
                    jz fehler_hinweis           ;nein, Fehler
                    mov ch, para1               ;1. Parameter holen
                    cmp ch, 13                  ;>13?
                    jg fehler_hinweis           ;ja, Fehler
                    mov cl, para2               ;2. Parameter holen
                    cmp cl, 13                  ;>13?
                    jg fehler_hinweis           ;ja, Fehler
                    cmp cl, ch                  ;1. Parameter > 2. Parameter
                    jl fehler_hinweis           ;ja, Fehler
                    jmp int                     ;sonst Wert aendern
fehler_hinweis:     push cs                     ;Datensegment korrigieren
                    pop ds
                    mov dx, offset fehler_text
                    mov ah, 9                   ;fehler_text ausgeben
                    int 21h
                    jmp dos                     ;Programm beenden
int:                mov ah, 1                   ;Funktion 1 einstellen
                    int 10h                     ;Wert aendern
                    jmp dos
cursor_normal:      mov ch, 12                  ;Normaleinstellung
                    mov cl, 13
                    mov ah, 1                   ;Funktion 1 einstellen
                    int 10h                     ;Wert aendern
dos:                mov ah, 4ch                 ;zurueck zum DOS
                    int 21h
            para1 db 0
            para2 db 0
fehler_text         db '?Fehlerhafter Parameter',13,'$'
                    code ends
                    end start
```

```
0B3A:0100  33 DB BF 80 00 8A 0D 32-ED 80 F9 00 75 03 E9 8C   3[?....2m.y.u.i.
0B3A:0110  00 47 8A 05 3C 20 74 52-3C 2C 75 07 FE C7 32 DB   .G..< tR<,u.~G2[
0B3A:0120  EB 48 90 3C 30 7C 63 3C-39 7F 5F 2C 30 80 FF 00   kH.<0|c<9._,0...
0B3A:0130  75 1D FE C3 80 FB 03 74-51 80 FB 01 74 0B 8A D0   u.~C.{.tQ.{.t..P
0B3A:0140  A0 A9 01 B6 0A F6 E6 02-C2 A2 A9 01 EB 1C 90 80    ).6.vf.B").k...
0B3A:0150  C3 01 80 FB 03 74 33 80-FB 01 74 0B 8A D0 A0 AA   C..{.t3.{.t..P *
0B3A:0160  01 B6 0A F6 E6 02 C2 A2-AA 01 E2 A5 80 FF 00 74   .6.vf.B"*.b%...t
0B3A:0170  19 8A 2E A9 01 80 FD 0D-7F 10 8A 0E AA 01 80 F9   ...)..}.....*..y

0B3A:0180  0D 7F 07 3A CD 7C 03 EB-0D 90 0E 1F BA AB 01 B4   ...:M|.k....:+.4
0B3A:0190  09 CD 21 EB 10 90 B4 01-CD 10 EB 09 90 B5 0C B1   .M!k..4.M.k..5.1
0B3A:01A0  0D B4 01 CD 10 B4 4C CD-21 00 00 3F 46 65 68 6C   .4.M.4LM!..?Fehl
0B3A:01B0  65 72 68 61 66 74 65 72-20 50 61 72 61 6D 65 74   erhafter Paramet
0B3A:01C0  65 72 0D 24 00 00 00 00-00 00 00 00 00 00 00 00   er.$............
```

Bild 2. Der Hex-Dump von CURSOR.COM

Neues vom Buchmarkt

Ein Anwenderhandbuch für den Commodore 128 hat der Verlag McGraw Hill herausgebracht. Auf 300 Seiten sind die wichtigsten Informationen zum Aufbau des Computers, seinen drei Betriebsarten und der Programmierung in Basic untergebracht. Das Buch wendet sich an Einsteiger und ist als Nachschlagewerk auch für Fortgeschrittene geeignet.

Peter Niemann, Das Commodore 128 Anwenderhandbuch, McGraw Hill, 1986, 300 Seiten, 40 DM. ISBN 3-89028-046-3

Dem Thema „Programmieren in Unix" wenden sich Günter Martin und Manfred Trostmann in ihrem jetzt im Vieweg-Verlag erschienenen Buch zu. Nach einer ausführlichen Beschreibung des Betriebssystems Unix-System V und vielen Hinweisen zur richtigen Bedienung werden Filter und die Shell als Programmiersprache erläutert. Der an Unix Interessierte erhält aus dem Buch nützliche Tips und kann nach der Lektüre Fehler finden und Laufzeitoptimierungen vornehmen.

Günter Martin, Manfred Trostmann, Programmieren mit Unix, Vieweg-Verlag, 1986, 220 Seiten, 36 DM. ISBN 3-528-04358-X

Seine jahrelangen Erfahrungen mit unterschiedlichen Unix-Systemen hat Eric Foxley in einem 270 Seiten starken Buch zusammengetragen. Mit viel Sachverstand werden die einzelnen Komponenten des Betriebssystems erklärt, wobei der Spezialist genauso wie der Anwender nützliche Informationen erhält. Ein umfangreiches Index-Verzeichnis und viele Beispiele machen das Buch zu einem wertvollen Nachschlagewerk.

Eric Foxley, Unix für Super-User, Addison-Wesley, 1987, 270 Seiten, 48 DM. ISBN 3-925118-24-1

Mit den Themen Robotik, Expertensysteme und Künstliche Intelligenz setzt sich Thomas Winzer in seinen Betrachtungen über die Perspektiven für eine künftige Welt auseinander. Durch 86 Abbildungen wird die Lektüre des Buches immer wieder aufgelockert und dadurch leicht verständlich. Der Autor spannt den weiten Bogen von den Grundzügen der Expertensysteme bis hin zum Selbstbau-Teach-In-Roboter.

Thomas Winzer, Künstliche Intelligenz und Robotik, Franzis-Verlag, 1987, 160 Seiten, 38 DM. ISBN 3-7723-8381-5

Wer sich mit der Programmiersprache C beschäftigen möchte, kann jetzt auf eine gelungene Einführung von Bruce H. Hunter zurückgreifen. Dabei ist das Buch für Anwender eines Unix-Systems ebenso geeignet wie für andere Betriebssysteme. Sehr viel Wert hat der Autor auf ausführliche Beispiele gelegt, die das Verständnis der oft komplexen Strukturen erleichtert.

Bruce H. Hunter, C – Eine Einführung, Sybex-Verlag, Düsseldorf, 1986, 293 Seiten, 48 DM. ISBN 3-88745-632-7

Mehr als eine Einführung in das Betriebssystem des Atari ST ist das Buch von Jürgen und Dieter Geiß, das sich an Software-Entwickler richtet. Alle Funktionen, die das komplexe System Atari ST zu bieten hat, werden beschrieben und in zwei umfangreichen Programmen beispielhaft angewandt. Dieses Buch ist nur etwas für Fortgeschrittene, die eigene GEM-Applikationen erstellen wollen.

Jürgen und Dieter Geiß, Softwareentwicklung auf dem Atari ST, Hüthig-Verlag, Heidelberg, 1986, 390 Seiten, 54 DM. ISBN 3-7785-1339-7

Markus Stecher

Druckertestroutine mit Hexcode-Anzeige

Hilfsprogramm zum Überprüfen von Druckerschnittstellen

Die meisten modernen Drucker bieten die Möglichkeit, Text als Hex-Code auszudrucken, um so ein Programm oder eine Schnittstelle zu testen. Am häufigsten benötigt man diese Funktion bei den Druckern, die sie standardmäßig nicht eingebaut haben. Dieses Programm für den IBM-PC und Kompatible stellt die nützliche Routine zur Verfügung.

Baut man selbst eine Schnittstelle oder versucht man, ein Textverarbeitungsprogramm an einen exotischen Drucker anzupassen, so möchte man wissen, welche Zeichen am Drucker ankommen und welche Steuerzeichen das Programm an den Drucker sendet.

Das Programm HexPrint (*Bild 1*) liefert diese Informationen. Es verbiegt den Interrupt-Vektor 17H, der für die Druckerausgabe zuständig ist, auf eine Routine, die von jedem Zeichen den Hex-Code auf dem Drucker ausgibt und das Zeichen nur ausdruckt, wenn es sich um ein druckbares Zeichen handelt (ASCII-Codes 32 bis 126). Ansonsten wird ein Punkt ausgegeben.

Auf einem 80-Zeichen-Drucker werden so 16 Zeichen pro Zeile ausgegeben. Gibt man beispielsweise unter DOS ein:

ECHO Dies ist ein Test.>PRN

```
code    segment
        assume cs:code
;
        org     0100h           ;Startadresse für COM-File
;
start:  jmp     newvekt         ;HexPrint initialisieren
;
intadr  equ     005Ch           ;Adresse des Interrupt-Vektor für INT 17
oldint  dd      0               ;Speicher für alten INT 17 - Vektor
chars   db      '0123456789ABCDEF' ;Hex-Ziffern
;
ausgabe: or     ah,ah           ;Zeichen-Ausgabe (AH=0, AL=Zeichen)
        jz      hexout
        jmp     far [oldint+2] ;nein, dann alte Routine
;
hexout: push    ds              ;Register sichern
        push    es
        push    dx
        push    cx
        push    bx
        push    ax
;
        mov     bx,0            ;alten Interrupt-Vektor wieder herstellen
        mov     es,bx
        mov     bx,[oldint]
        mov     cx,[oldint+2]
        mov     es:[intadr  ],bx
        mov     es:[intadr+2],cx
;
        mov     bx,ax           ;Zeichen in Hex umwandeln
        mov     cl,4
        rol     bx,cl           ;High Nibble -> BH
        mov     bl,bh
        xor     bh,bh           ;High Nibble -> BX
        mov     cx,ax
        mov     al,chars[bx] ;1. Hex-Zeichen -> AL
        int     17h             ;1. Zeichen ausgeben
;
        mov     bx,cx
        and     bl,00001111b ;Low Nibble -> BL
        mov     al,chars[bx] ;2. Hexzeichen -> AL
        xor     ah,ah
        int     17h             ;2. Zeichen ausgeben
;
        mov     al,' '          ;Leerzeichen ausgeben
        xor     ah,ah
        int     17h
;
        mov     ax,cx           ;Zeichen selber ausgeben
        cmp     al,' '          ;Zeichen nicht druckbar => .
        jge     noprt1
        mov     al,'.'
noprt1: cmp     al,126
        jle     noprt2
        mov     al,'.'
noprt2: int     17h             ;Zeichen bzw. '.' ausgeben
;
        mov     al,' '          ;Leerzeichen ausgeben
        xor     ah,ah
        int     17h
;
        lea     bx,ausgabe      ;neuen Interrupt-Vektor wieder herstellen
        mov     es:[intadr  ],bx
        mov     es:[intadr+2],cs
```

Bild 1. DOS-Interrupt 17H wird so umgeleitet, daß statt des Zeichens dessen Hex-Code ausgegeben wird

```
;
        pop     ax              ;Register wieder herstellen
        pop     bx
        pop     cx
        pop     dx
        pop     es
        pop     ds
;
        iret                    ;Rückkehr vom Interrupt
;
;
newvekt: mov    bx,0            ;alten Interrupt-Vektor holen
        mov     es,bx
        mov     bx,es:[intadr]
        mov     cx,es:[intadr+2]
        mov     oldint,bx       ;und bei oldint speichern
        mov     oldint+2,cx
;
        lea     bx,ausgabe      ;neuen Interrupt-Vektor setzen
        mov     es:[intadr  ],bx
        mov     es:[intadr+2],cs
;
        lea     dx,newvekt      ;Speicherplatz markieren
        int     27h             ;und Programm beenden
code    ends
        end     start
```

```
0B39:0100  E9 8E 00 00 00 00 00 30-31 32 33 34 35 36 37 38   i......012345678
0B39:0110  39 41 42 43 44 45 46 0A-E4 74 05 2E FF 2E 03 01   9ABCDEF.dt......
0B39:0120  1E 06 52 51 53 50 BB 00-00 8E C3 2E 8B 1E 03 01   ..RQSP;...C.....
0B39:0130  2E 8B 0E 05 01 26 89 1E-5C 00 26 89 0E 5E 00 8B   .....&..\.&..^..
0B39:0140  D8 B1 04 D3 C3 8A DF 32-FF 8B C8 2E 8A 87 07 01   X1.SC._2..H.....
0B39:0150  CD 17 8B D9 80 E3 0F 2E-8A 87 07 01 32 E4 CD 17   M..Y.c......2dM.
0B39:0160  B0 20 32 E4 CD 17 8B C1-3C 20 7D 02 B0 2E 3C 7E   0 2dM..A< ).0.<~
0B39:0170  7E 02 B0 2E CD 17 B0 20-32 E4 CD 17 8D 1E 17 01   ~.0.M.0 2dM.....
0B39:0180  26 89 1E 5C 00 26 8C 0E-5E 00 58 5B 59 5A 07 1F   &..\.&..^.X[YZ..
0B39:0190  CF BB 00 00 8E C3 26 8B-1E 5C 00 26 8B 0E 5E 00   O;...C&..\.&..^.
0B39:01A0  2E 89 1E 03 01 2E 89 0E-05 01 8D 1E 17 01 26 89   ..............&.
0B39:01B0  1E 5C 00 26 8C 0E 5E 00-8D 16 91 01 CD 27 00 00   .\.&..^.....M'..
```

Bild 2. Das Druckertestprogramm in Hex-Dump-Darstellung

so erhält man auf dem Drucker diese Ausgabe:
44 D 69 i 65 e 73 s 20 69 i 73 s 74 t 20 65 e 69 i 6E n 20 54 T 65 e 73 s 74 t 2E . 0D . 0A .
0D bedeutet Wagenrücklauf (Carriage Return) und 0A Zeilenvorschub (Line Feed).
HexPrint läßt sich nur durch Booten wieder ausschalten.
Mit DEBUG läßt sich das Programm (*Bild 2*) auch ohne Assembler eingeben.

Voreinstellungen bei dBaseII ändern

Die Belegung der Funktionstasten in dBaseII läßt sich normalerweise nur nach Aufruf von dBase mit dem SET-Befehl ändern. Hier wird ein Patch beschrieben, der nach dem Aufruf von dBase die Funktionstasten nach Ihren Wünschen belegt.
Dieser Tip ist für die PC-DOS-Version von dBaseII 2.4 geschrieben.
Die in *Tabelle 1* angegebenen Byte-Adressen setzen die SET-Schalter. Auch hier gilt 00 für OFF und 256 oder FFH für ON.

Tabelle 1. ▶
Die Adressen der SET-Schalter

SET-Befehl	Adresse HEX	Adresse DEC
CARRY	03C2	962
CONFIRM	03C6	966
DEBUG	03C7	967
ECHO	03DC	988
ESCAPE	03DE	990
INTENSITY	03E3	995
STEP	03E9	1001
BELL	03BF	959
COLON	03C5	965
DELETE	03C9	969
EJECT	03E8	1000
EXACT	03DF	991
LINKAGE	03E6	998
RAW	03E7	999
TALK	03EB	1003

Auch die Funktionstasten lassen sich so neu setzen. Der Text darf maximal 31 Byte lang sein und beginnt auf der angegebenen Adresse (*Tabelle 2*). Folgt dem Text ein Semikolon, wird der Befehl sofort nach Betätigen der Funktionstaste ausgeführt. Ohne dieses Semikolon muß zur Ausführung ‹RETURN› gedrückt werden.

Tabelle 2.
Die Adressen der Funktionstasten

F-Taste	Adresse HEX	Adresse DEC
F1	0418	1048
F2	0437	1079
F3	0456	1110
F4	0475	1141
F5	0494	1172
F6	04B3	1203
F7	04D2	1234
F8	04F1	1265
F9	0510	1296
F10	052F	1327

Für alle, die mit DEBUG nicht so geübt sind, je ein Beispiel für einen SET-Befehl und eine Funktionstasten-Belegung. Kopieren sie die Datei dBase.com in zum Beispiel db.com. Wir wollen beim Aufruf von dBase „SET COLON OFF“ gesetzt haben.
Die Eingaben, die Sie machen müssen, sind in Anführungszeichen gesetzt.
„debug db.com“
–„E 3C5“ RETURN
XXXX:03C5 FF.„00“ RETURN (XXXX ist abhängig vom RAM-Speicher)
–„W“ RETURN
Schreiben von 7E00 Byte
–„Q“ RETURN
Jetzt können Sie DB eingeben und den SET-Befehl überprüfen.
Die Funktionstaste F9 soll das Kommando „MODIFY COMMAND“ erhalten:

„debug db.com“
–„E DS:510 F3 „MODIFY COMMAND“ RETURN (F3 ist Funktionstaste 3)
–„W“ RETURN
Schreiben von 7E00 Byte
–„Q“ RETURN
Es muß beachtet werden, daß der neue Text den alten überschreiben muß, d. h. er muß gleich lang oder länger sein, damit keine Reste des alten Textes stehenbleiben; evtl. mit Leertaste auffüllen.
Noch ein letzter Hinweis:
Wenn Sie Ansi.sys geladen und mit dem DOS-Befehl „PROMPT“ die Funktionstasten belegt haben, erscheinen diese auch nach dem Aufruf von dBase. dBase kann mit zum Beispiel „DIR/P“ nichts anfangen.

Dipl.-Ing. Hartwig Boehm

Stefan Wurdack

Der mc-68-ECB und CP/M-68k

Teil 3

Der dritte Teil dieser Artikelserie über den mc-68-ECB behandelt die Installation des CP/M-68k. Dieses 16-Bit-Betriebssystem von Digital Research ist sehr stark an seine 8-Bit-Vorgänger angelehnt. Die Struktur der Datenverwaltung auf den Massenspeichern ist exakt gleich. Diese Übereinstimmung gestattet es auf sehr einfach Art und Weise, die meisten Aufgaben des 16-Bit-BIOS (Basic Input/Output System) mit dem vorhandenen 8-Bit-BIOS zu erfüllen.

Neben den üblichen Standardprogrammen zur Software-Entwicklung enthält der Lieferumfang von CP/M-68k auch noch einen kompletten C-Compiler. Die gemeinsame Verwendung der gleichen Speichermedien durch beide CPUs bringt unschätzbare Vorteile mit sich.

Zum einen kann durch den einfachen Aufruf von entsprechenden Programmen das 16-Bit-Betriebssystem betreten oder verlassen werden, ohne daß Disketten gewechselt werden müßten, da 8- und 16-Bit-Dateien beliebig nebeneinander auf den Disketten abgespeichert werden können. Dadurch lassen sich zur Programmentwicklung die vorhandenen Editoren einsetzen und die erarbeiteten Programme anschließend mit dem CP/M-68k-Assembler oder C-Compiler übersetzen. Zum anderen stellt die CPU 68000 eine für Z80-Verhältnisse gewaltige Rechenleistung zur Verfügung, die völlig unabhängig parallel zu Z80-Programmen eingesetzt werden kann. Das bedeutet, daß zwei Prozessoren gleichzeitig an der Lösung eines Problems arbeiten können, und dann dabei völlig unabhängig voneinander verschiedene Teilbereiche bearbeiten. Durch geschickte Programmierung lassen sich die Ergebnisse miteinander verknüpfen und eine wesentliche Reduktion der Programmlaufzeit erreichen . Ein Beispiel dafür wäre eine bewegte Grafik, bei der der Z80 den Bildaufbau steuert, während der 68000 die Koordinaten für die Darstellung errechnet.

Das 16-Bit-BIOS

Jedem, der sich schon einmal näher mit CP/M befaßt hat, ist der Begriff BIOS klar. Es handelt sich um die Schnittstelle zwischen der „logischen" Welt des Betriebssystems und dem physikalischen Aufbau des Rechners. Alle Daten, die an Sichtgeräte, Drucker, Floppies ... ausgegeben bzw. von dort geholt werden, müssen über bestimmte Kanäle geleitet werden. Da diese Kanäle von Rechner zu Rechner verschieden konstruiert sind, muß jedes Betriebssystem, das auf Maschinen unterschiedlichen Aufbaus laufen soll, einen solchen Programmteil besitzen, der diese Kanäle individuell, abhängig von der Hardware bedient. Da der mc-68-ECB aber auf jedem DMA-fähigen CP/M-8-Bit-System laufen soll, ohne daß ein Anwender erst das BIOS an seine Hardwarekonfiguration anpassen müßte, ist die Installation des BIOS im 68000-System mit etwas Aufwand verbunden. Die Hauptschwierigkeit besteht darin, dem BIOS *(Bild 1)* die notwendigen Parameter der Massenspeichereinheiten zugänglich zu machen. Die wesentlichen Informationen über die vorhandenen Speichereinrichtungen sind in den DPHs (Disk Parameter Header) und DPBs (Disk Parameter Block, *Bild 2*) auf der Z80-Seite abgelegt. Diese Informationen müssen in der Initialisierungsroutine (allg. Kaltstart) des 16-Bit-BIOS erst aus dem 8-Bit-BIOS kopiert werden, bevor ein Zugriff auf Floppies oder andere Einrichtungen möglich ist. Die genauen Abläufe innerhalb der Initialisierungsroutine werden am Ende des Artikels eingehend dargestellt, da sie den mc-68-ECB völlig unabhängig von der Hardware des Hostrechners machen und damit dessen universelle Verwendbarkeit garantieren.

Die BIOS-Funktionen bei CP/M-68k

Die vertrauten BIOS-Routinen Nr. 0...16 im 8-Bit-CP/M sind auch unter CP/M-68k vorhanden. Zusätzlich gibt es noch Funktionen 18...22, deren Aufgaben weiter unten noch genauer erläutert werden. Bei allen BIOS-Zugriffen haben die einzelnen CPU-Register des 68000 folgende Funktionen: Register D0.W enthält die Nummer der gewünschten BIOS-Routine, D1 und D2 die notwendigen Parameter. Sämtliche Aufrufe des BIOS durch das BDOS erfolgen über den Softwareinterrupt Trap #3. Die Ergebnisse werden vom BIOS, soweit vorhanden, im Register D0 ans BDOS übergeben. Für das 8-Bit-BIOS müssen die Funktionsnummern beim Aufruf im Register C und die eventuell notwendigen Parameter in BC und DE stehen. Ergebnisse werden vom 8-Bit-BIOS im Register A geliefert, vom 16-Bit-BIOS im Register HL. Der Monitortreiber, der bereits in der letzten MC vorgestellt wurde, ist für alle BIOS-Zugriffe verwendbar, da er sämtliche parametertragenden Register in das Datenfeld kopiert, so daß sie ohne weiteres vom BIOS des 68000 gelesen werden können. Im folgenden soll nun kurz auf die einzelnen BIOS-Funktionen eingegangen werden.

Der Kaltstart (0) stellt für den mc-68-ECB eine sehr wichtige Funktion dar, da dieses Programm sämtliche Laufwerksparametertabellen aufbaut und gleichzeitig die Größe der TPA (Transparent Program Area) dynamisch an die Größe der installierten RAM-Floppy anpaßt. Bei einem Warmstart (1) springt der Prozessor lediglich den CCP (Console Command Processor) an. Da dieser resident im Speicher bleibt und nicht wie unter CP/M 2.2 von manchen Programmen überschrieben wird, meldet sich das Betriebssystem sofort wieder mit seinem Prompt, ohne das Boot-Laufwerk in Betrieb zu nehmen. Die BIOS-Funktionen Konsolenstatus (2), Konsoleneingabe (3), Lochstreifenleser (7) und Druckerstatus (15) werden von einem Unterprogramm

```
 1
 2
 3                              ********************************************************************
 4                              *                                                                  *
 5                              *                    CP/M-68K BIOS                                 *
 6                              *              Für MC-68-ECB CPM-System   12.6.86                  *
 7                              *                                                                  *
 8                              ********************************************************************
 9                              * Laufwerksnummer für Ramfloppy muß an Adresse 0 stehen
10                              globl _init,_ccp
11
12                              slatch  =       $600001
13                              zlat    =       $200008 * EPROM-Adresse $20000B enthält I/O-Adresse des
14                              solflo  =       $20000C * Start des Ramfloppytreibers
15                              *                       Systemlatchs ( schreiben )
16                              z80mem  =       $500001 * Beginn des Z80 Adressraums für 68000
17                              iobyte  =       z80mem+2*$3     * IO-Byte im Z80-Raum
18                              combyte =       2*$103+z80mem   * Kommandobyte
19                              ready   =       combyte+2       * Ready-Byte
20                              data    =       combyte+4       * Beginn des Parameterübergabefelds
21                              readyz80        =       $ff     * "Fertig"-Meldung von Z80
22                              startz80        =       $ff     * Start-Zeichen an Z80
23
24                              memflo  =       0       * Ramfloppy-Laufwerksnumer steht an Adresse 0
25                              *                         Muß bei der Ramfloppyinitialisierung angegeben
26                              *                         werden.
27                              begtpa  =       $8000   * Beginn der TPA oberhalb des CP/M 68k
28                              xseldsk =       9       * Nummer der Biosfunktion zur Anwahl eines Laufwerks
29                              dphlen  =       26      * Länge eines dph's
30
31                              *       Initialisierungsroutine für das System nach einem Kaltstart
32                              *       bzw. Wiedereintritt aus dem Z80-CP/M
33
34 00000000 267C00000080       _init:  movea.l #distable,a3            * Automatischer Aufbau der DPB's
35 00000006 323C000F                   move.w  #16-1,d1                * durch Auslesen der Informationen
36 0000000A 16FC00FF           init1:  move.b  #$ff,(a3)+              * aus dem Z80-Bios
37 0000000E 421B                       clr.b   (a3)+
38 00000010 51C9FFF8                   dbf     d1,init1
39 00000014 267C00000080               movea.l #distable,a3            * Tabelle zum Umsetzen der Laufwerks-
40                              *                                        nummern.
41 0000001A 287C000000A0               movea.l #dph0,a4                * Anfang der Disk Parameter Header (DPH)
42 00000020 4244                       clr.w   d4
43 00000022 4287                       clr.l   d7
44 00000024 4282                       clr.l   d2
45 00000026 6100013C           init2:  bsr     tstdsk          * Laufwerk vorhanden ? Ein: d7 outp: d2 dphadr.
46 0000002A 5247                       addq.w  #1,d7           * Laufwerksnummer erhöhen
47 0000002C 61000174                   bsr     copydpar        * Laufwerksparameter kopieren
48 00000030 0C070010                   cmpi.b  #16,d7          * Laufwerke 0 - 15
49 00000034 66F0                       bne     init2
50 00000036 4287                       clr.l   d7              * Laufwerkszähler auf null setzen
51 00000038 610001B0           init3:  bsr     copydpb         * DPB kopieren
52 0000003C 5207                       addq.b  #1,d7           * Laufwerkszähler erhöhen
53 0000003E 0C070010                   cmpi.b  #16,d7          * Alle Laufwerke ?
54 00000042 66F4                       bne     init3           *
55 00000044 4287                       clr.l   d7              * Laufwerkszähler löschen
56 00000046 3207               init4:  move.w  d7,d1
57 00000048 61000186                   bsr     getintdrv       * Bios-interne Laufwerksnummer holen
58 0000004C 5207                       addi.b  #1,d7
59 0000004E 0C0000FF                   cmpi.b  #$ff,d0         * Laufwerk vorhanden ?
60 00000052 6714                       beq     init5
61 00000054 610000F8                   bsr     getdph
62 00000058 2640                       move.l  d0,a3           * dph Adresse
63 0000005A 2441                       move.l  d1,a2           * dpb Adresse
64 0000005C 274C0012                   move.l  a4,18(a3)       * Schreibe CV (Check Vector)  Adresse in dph
65 00000060 4281                       clr.l   d1
66 00000062 322A000C                   move.w  12(a2),d1       * hole CV Größe
67 00000066 D9C1                       adda.l  d1,a4           * Nächste freie Speicheradressse
68 00000068 0C070010           init5:  cmpi.b  #16,d7          * fertig ?
69 0000006C 66D8                       bne     init4
70 0000006E 4287                       clr.l   d7
71                             init6:
72 00000070 3207                       move.w  d7,d1
73 00000072 6100015C                   bsr     getintdrv       * Interne Laufwerksnummer holen
74 00000076 5207                       addi.b  #1,d7
75 00000078 0C0000FF                   cmpi.b  #$ff,d0         * Laufwerk vorhanden ?
76 0000007C 6720                       beq     init7
77 0000007E 610000CE                   bsr     getdph
78 00000082 2640                       move.l  d0,a3           * dph Adresse
79 00000084 2441                       move.l  d1,a2           * dpb Adresse
80 00000086 274C0016                   move.l  a4,22(a3)       * Schreibe AV (Alloc. Vector) Adresse in dph
81 0000008A 4281                       clr.l   d1
82 0000008C 322A0006                   move.w  6(a2),d1        * Hole Anzahl der Blöcke -1 aus dpb
83 00000090 82FC0008                   divu    #8,d1
84 00000094 5241                       addq.w  #1,d1           * Berechne die AV Größe
85 00000096 4841                       swap    d1
86 00000098 4241                       clr.w   d1              * lösche Rest
87 0000009A 4841                       swap    d1
88 0000009C D9C1                       adda.l  d1,a4           * Nächste freie Speicheradresse
89 0000009E 0C070010           init7:  cmpi.b  #16,d7          * fertig ?
90 000000A2 66CC                       bne     init6
91 000000A4 23FC000002460000008C       move.l  #traphndl,$8c   * trap #3 handler initialisieren
92 000000AE 303C0009                   move.w  #xseldsk,d0
93 000000B2 123900000000               move.b  memflo,d1
94 000000B8 4E43                       trap    #3              * dph für Ramfloppy holen
95 000000BA C188                       exg     a0,d0
96 000000BC 2068000E                   movea.l 14(a0),a0       * dpb Adresse in a0
97 000000C0 3028000E                   move.w  14(a0),d0       * Track-Offset in d0.w
98 000000C4 C0FC2000                   mulu    #$2000,d0       * Ende der TPA berechnen
99 000000C8 2200                       move.l  d0,d1           * Adresse merken
100 000000CA 5381                      subq.l  #1,d1           * letztes nutzbares Byte
101 000000CC 90BC00008000              sub.l   #begtpa,d0      * Länge der TPA
102 000000D2 632A                      bls     init10          * Systemabbruch
103 000000D4 23C000000010              move.l  d0,memrgn+6     * Länge in Tabelle eintragen
104 000000DA 207C000000FF              movea.l #tpa1,a0        * 1. Puffer für Wandlung
105 000000E0 203C00008000              move.l  #begtpa,d0
106 000000E6 6124                      bsr     outadr
107 000000E8 207C00000109              movea.l #tpa2,a0
108 000000EE 2001                      move.l  d1,d0
109 000000F0 611A                      bsr     outadr
110 000000F2 207C00000054              move.l  #cbootmsg,a0    * Kaltstartmeldung ausgeben
111 000000F8 613C                      bsr     outstr
112 000000FA 42B0                      clr.l   d0              * disk A, user 0 aktivieren
113 000000FC 4E75                      rts
114
115                            init10:
116 000000FE 207C00000014              move.l  #initerr,a0
```

Bild 1. Das BIOS auf der 68000-Seite

```
117 00000104 6130                       bsr     outstr
118 00000106 4EF90020000C               jmp     solflo
119
120                             *       binaer-ascii Konversion
121                             *       Zahl mit 2-8 Digits wird gewandelt ( Reg. D0 )
122                             *       und im Puffer (A0) gespeichert
123
124 0000010C 4840               outadr: swap    d0
125 0000010E 610A                       bsr     outnb           * a0 - a19 valid
126 00000110 4840                       swap    d0
127 00000112 2F00               outnw:  move.l  d0, -(a7)
128 00000114 E058                       ror.w   #8, d0
129 00000116 6102                       bsr     outnb
130 00000118 201F                       move.l  (a7)+, d0
131 0000011A 2F00               outnb:  move.l  d0, -(a7)
132 0000011C E800                       asr.b   #4, d0
133 0000011E 6102                       bsr     outn
134 00000120 201F                       move.l  (a7)+, d0
135 00000122 C03C000F           outn:   and.b   #$f,d0
136 00000126 B03C0030                   or.b    #'0',d0
137 0000012A B03C0039                   cmp.b   #'9',d0
138 0000012E 6F02                       ble     outn1
139 00000130 5E00                       add.b   #7,d0
140 00000132 10C0               outn1:  move.b  d0,(a0)+  * char. to buffer
141 00000134 4E75                       rts
142
143                             outstr:  * ein: a0 deutet auf  1.Byte des Strings, der mit 0 enden muß.
144                                      * aus: a0 = Adresse des Abschlußzeichens ( 0 ) + 1
145
146 00000136 48E7C000                   movem.l d0/d1,-(a7)
147 0000013A 121B               outslp: move.b (a0)+,d1
148 0000013C 670A                       beq     outs1
149 0000013E 303C0010                   move.w  #(4*4),d0               *  Bios Konsolen-Ausgabe * 4
150 00000142 610002A2                   bsr     bz80
151 00000146 60F2                       bra     outslp
152 00000148 4CDF0003           outs1:  movem.l (a7)+,d0/d1
153 0000014C 4E75                       rts
154
155
156                             getdph:         * ein : d0.l interne Laufwerksnummer
157                                             * aus : d0 dph Adr. : d1 dpb Adr.
158 0000014E 2F08                       move.l  a0,-(a7)
159 00000150 C0FC001A                   mulu.w  #dphlen,d0
160 00000154 D0BC000000A0              add.l   #dph0,d0        * dph Adr.
161 0000015A 2040                       move.l  d0,a0
162 0000015C 2228000E                   move.l  14(a0),d1       * dpb Adr.
163 00000160 205F                       move.l  (a7)+,a0
164 00000162 4E75                       rts
165
166                             *       ein : d7 Laufwerk
167                             *       aus : d2 dph Adresse im Z80-Raum ( 0 wenn Laufwerk nicht vorhanden )
168
169 00000164 303C0009           tstdsk: move.w  #xseldsk,d0
170 00000168 E548                       lsl.w   #2,d0
171 0000016A 3207                       move.w  d7,d1           * kopiere Laufwerk in D1
172 0000016C 61000278                   bsr     bz80            * hole dph Adresse von Z80
173 00000170 207C0050020F               movea.l #(data+4),a0
174 00000176 1410                       move.b  (a0),d2
175 00000178 E14A                       lsl.w   #8,d2
176 0000017A 14280002                   move.b  2(a0),d2        * dph Adresse im z80-Adressraum
177 0000017E 4E75                       rts
178
179                             *       Byte aus Z80 Raum an der Adresse a0 lesen ( a0 = 0 bis $ffff )
180                             *       ein : a0
181                             *       aus : Byte in d0.b : a0 = a0 + 1
182 00000180 2F08               rz80byte:       move.l  a0,-(a7)
183 00000182 D1C8                       adda.l  a0,a0
184 00000184 D1FC00500001               adda.l  #z80mem,a0
185 0000018A 1010                       move.b  (a0),d0
186 0000018C 205F                       move.l  (a7)+,a0
187 0000018E 5288                       addq.l  #1,a0
188 00000190 4E75                       rts
189
190                             *       Z80 Wort lesen
191                             *       ein : a0 ( 0 bis $ffff )
192                             *       aus : d0.w : a0 = a0 + 2
193
194                             rz80word:
195 00000192 2F08                       move.l  a0,-(a7)
196 00000194 5288                       addq.l  #1,a0
197 00000196 61E8                       bsr     rz80byte
198 00000198 205F                       move.l  (a7)+,a0
199 0000019A E148                       lsl.w   #8,d0
200 0000019C 61E2                       bsr     rz80byte
201 0000019E 5288                       addq.l  #1,a0           * Adresszähler erhöhen
202 000001A0 4E75                       rts
203
204                             *       dph im 68000-Raum aufbauen und die notwendigen Informationen
205                             *       aus dem Z80-Raum kopieren
206                             *       ein : d2 dph Adresse im Z80 Raum : a3 zeigt auf Laufwerksliste
207                             *             a4 dph Adresse im 68000 Raum
208                             copydpar:
209 000001A2 4A42                       tst.w   d2              * d2.w <> 0 Laufwerk vorhanden
210 000001A4 6726                       beq     copyex
211 000001A6 421B                       clr.b   (a3)+           * trage Laufwerk als vorhanden in Liste ein
212 000001A8 16C4                       move.b  d4,(a3)+        * vergebe interne Laufwerksnummer
213 000001AA 5204                       addq.b  #1,d4           * interne Nummer erhöhen
214 000001AC 4280                       clr.l   d0
215 000001AE 2042                       move.l  d2,a0           * dph Adress Z80
216 000001B0 61E0                       bsr     rz80word        * hole Adresse für Übersetzungstafel im Z80 Raum
217 000001B2 28C0                       move.l  d0,(a4)+        * im 68000 dph speichern
218 000001B4 425C                       clr.w   (a4)+           * Scratch Memory
219 000001B6 425C                       clr.w   (a4)+
220 000001B8 425C                       clr.w   (a4)+
221 000001BA 28FC00000000               move.l  #dirbuf,(a4)+   * Adresse für Dirbuf einsetzen
222 000001C0 5088                       addq.l  #8,a0
223 000001C2 61CE                       bsr     rz80word        * dpb Adresse im Z80 Raum holen
224 000001C4 28C0                       move.l  d0,(a4)+        * vorerst im 68000 dph speichern
225 000001C6 429C                       clr.l   (a4)+
226 000001C8 429C                       clr.l   (a4)+           * a4 zeigt auf den nächsten dph
227 000001CA 4E75                       rts
228 000001CC 548B               copyex: addq.l  #2,a3           * kein Eintrag in die Laufwerksliste
229 000001CE 4E75                       rts
230
231                             *       Interne Laufwerksnummer aus der Liste lesen
232                             *       ein : d1 externe Laufwerksnummer
```

```
000001D0 2F08                   *       aus : d0 interne Laufwerksnummer oder $ff wenn nicht vorhanden
                                getintdrv:
000001D0 2F08                           move.l  a0,-(a7)
000001D2 207C00000080                   movea.l #distable,a0
000001D8 4280                           clr.l   d0
000001DA 1001                           move.b  d1,d0
000001DC E308                           lsl.b   #1,d0
000001DE D1C0                           adda.l  d0,a0           * Adresse aus Laufwerksnummer berechnen
000001E0 1018                           move.b  (a0)+,d0        * Laufwerk vorhanden ?
000001E2 6602                           bne     gdex            * nein --> Sprung
000001E4 1010                           move.b  (a0),d0         * interne  Nummer
000001E6 205F                   gdex:   move.l  (a7)+,a0
000001E8 4E75                           rts

                                *       dpb kopieren
                                *       ein : d7 externe Laufwerksnummer ; a4 dpb Startadresse 68000 Raum
                                *       aus : a4 deutet auf den Beginn des nächsten dpb
                                copydpb:
000001EA 3207                           move.w  d7,d1
000001EC 61E2                           bsr     getintdrv
000001EE 0C0000FF                       cmpi.b  #$ff,d0
000001F2 6750                           beq     dpbex           * nicht vorhanden --> Sprung
000001F4 267C000000A0                   movea.l #dph0,a3        * Adresse für dph0 holen
000001FA C0FC001A                       mulu.w  #dphlen,d0      * Adresse des aktuellen dph's berechnen
000001FE D7C0                           adda.l  d0,a3           *
00000200 206B000E                       movea.l 14(a3),a0       * hole z80 dpb Adresse aus 68000 dph
00000204 274C000E                       move.l  a4,14(a3)       * schreibe 68000 dpb Adresse in dph
00000208 6188                           bsr     rz80word        * Kopiere alle notwendigen Parameter
0000020A 38C0                           move.w  d0,(a4)+
0000020C 6100FF72                       bsr     rz80byte
00000210 18C0                           move.b  d0,(a4)+
00000212 6100FF6C                       bsr     rz80byte
00000216 18C0                           move.b  d0,(a4)+
00000218 6100FF66                       bsr     rz80byte
0000021C 18C0                           move.b  d0,(a4)+
0000021E 421C                           clr.b   (a4)+           * Dummy-Bbyte einfügen
00000220 6100FF70                       bsr     rz80word
00000224 38C0                           move.w  d0,(a4)+
00000226 6100FF6A                       bsr     rz80word
0000022A 38C0                           move.w  d0,(a4)+
0000022C 6100FF52                       bsr     rz80byte
00000230 18C0                           move.b  d0,(a4)+
00000232 6100FF4C                       bsr     rz80byte
00000236 18C0                           move.b  d0,(a4)+
00000238 6100FF58                       bsr     rz80word
0000023C 38C0                           move.w  d0,(a4)+
0000023E 6100FF52                       bsr     rz80word
00000242 38C0                           move.w  d0,(a4)+
00000244 4E75                   dpbex:  rts

                                *       Trap-Handler bei Biosaufrufen

                                traphndl:
00000246 0C400017                       cmpi    #nfuncs,d0
0000024A 6408                           bcc     trapng
0000024C E548                           lsl     #2,d0           * Bios Funktionsnummer * 4
0000024E 207B0006                       movea.l 6(pc,d0),a0     * hole Programmadresse
00000252 4E90                           jsr     (a0)            * entsprechendes Unterprogramm anspringen

                                trapng:
00000254 4E73                           rte

                                *               Tabelle mit Einsprungadressen
                                biosbase:

00000256 00000000                       .dc.l   _init
0000025A 000002B2                       .dc.l   wboot
0000025E 000002B8                       .dc.l   constat
00000262 000002B8                       .dc.l   conin
00000266 000002C6                       .dc.l   conout
0000026A 000002C6                       .dc.l   lstout
0000026E 000002C6                       .dc.l   pun
00000272 000002B8                       .dc.l   rdr
00000276 000002C6                       .dc.l   home
0000027A 000002CC                       .dc.l   seldsk
0000027E 000002F6                       .dc.l   settrk
00000282 000002FE                       .dc.l   setsec
00000286 0000031A                       .dc.l   setdma
0000028A 00000322                       .dc.l   read
0000028E 00000396                       .dc.l   write
00000292 000002B8                       .dc.l   liststt
00000296 00000306                       .dc.l   sectran
0000029A 0000031A                       .dc.l   setdma
0000029E 00000432                       .dc.l   getseg
000002A2 0000043A                       .dc.l   getiob
000002A6 00000444                       .dc.l   setiob
000002AA 0000042E                       .dc.l   flush
000002AE 0000044C                       .dc.l   setexc

                                        nfuncs=(*-biosbase)/4

000002B2 4EF900000000           wboot:  jmp     _ccp    * Nur CCP anspringen


                                liststt:
                                rdr:
                                conin:
000002B8 6100012C               constat:bsr     bz80
000002BC 4240                           clr.w   d0
000002BE 103900050020D                  move.b  data+2,d0
000002C4 4E75                           rts

                                conout  equ     *
                                lstout  equ     *
                                pun     equ     *
000002C6 6100011E               home:   bsr     bz80
000002CA 4E75                           rts

                                *       Laufwerk auswählen
                                *       ein : Laufwerk in d1.b
                                *       aus : dph Adresse in d0.l
                                seldsk:
000002CC 13C100000000                   move.b  d1,seldrv       * speichere Laufwerksnummer
000002D2 267C00000080                   movea.l #distable,a3
000002D8 02810000000F                   andi.l  #$f,d1
000002DE 4280                           clr.l   d0
000002E0 E309                           lsl.b   #1,d1           * *2
```

```
000002E2 D7C1                   adda.l  d1,a3           * Laufwerkstabellen-Eintritt berechnen
000002E4 4A1B                   tst.b   (a3)+          * Laufwerk vorhanden ?
000002E6 660C                   bne     selrtn         * nein --> Sprung
000002E8 1013                   move.b  (a3),d0        * hole interne Laufwerksnummer
000002EA C0FC001A               mulu    #dphlen,d0
000002EE D0BC000000A0           add.l   #dph0,d0       * d0 zeigt auf zugehörigen dph
000002F4 4E75           selrtn: rts

                        settrk: move.w  d1,track
000002F6 33C100000002           rts
000002FC 4E75

                        setsec: move.w  d1,sector
00000304 4E75                   rts

                        sectran:


00000306 610000DE               bsr     bz80           * d2.w = Adresse der Sektorübers.tab. im z80
0000030A 103900500200           move.b  data+4,d0      * hole physikalische Sektornummer
00000310 E148                   lsl.w   #8,d0
00000312 103900500211           move.b  data+6,d0
00000318 4E75                   rts

                        setdma:
0000031A 23C100000006           move.l  d1,dma
00000320 4E75                   rts

                        *       Laufwerks-Leseroutine

00000322 123900000000   read:   move.b  memflo,d1
00000328 143900000000           move.b  seldrv,d2
0000032E B401                   cmp.b   d1,d2          * Laufwerk = Ramfloppy ?
00000330 6738                   beq     memread        * ja --> Sprung auf speziellen Treiber
00000332 323900000002           move.w  track,d1
00000338 343900000004           move.w  sector,d2
0000033E 163900000000           move.b  seldrv,d3
00000344 610000A0               bsr     bz80           * Alle Parameter an Z80 übergeben
00000348 10390050020D           move.b  data+2,d0
0000034E 6618                   bne     error          * BIOS Fehlermeldung auswerten
00000350 207C00500215           move.l  #(data+10),a0  * Adresse des Sektorbuffers im Z80 Raum
00000356 227900000006           move.l  dma,a1
0000035C 363C007F               move.w  #127,d3
00000360 12D0           read1:  move.b  (a0),(a1)+     * Daten holen
00000362 5488                   addq.l  #2,a0
00000364 51CBFFFA               dbf     d3,read1
00000368 4E75           error:  rts

                        *       Ramfloppy Leseroutine
                        memread:
0000036A 6114                   bsr     getad          * Adresse aus Track und Sektornummer berechnen
0000036C 227900000006           move.l  dma,a1
                        datamove:
00000372 363C007F               move.w  #127,d3
00000376 12D8           data1:  move.b  (a0)+,(a1)+
00000378 51CBFFFC               dbf     d3,data1
0000037C 4240                   clr.w   d0             * alles o.k.
0000037E 4E75                   rts

00000380 4283           getad:  clr.l   d3
00000382 363900000002           move.w  träck,d3
00000388 ED8B                   lsl.l   #6,d3
0000038A D67900000004           add.w   sector,d3
00000390 EF8B                   lsl.l   #7,d3
00000392 2043                   move.l  d3,a0          * Adresse berechnet
00000394 4E75                   rts

00000396 123900000000   write:  move.b  memflo,d1
0000039C 143900000000           move.b  seldrv,d2
000003A2 B401                   cmp.b   d1,d2
000003A4 6734                   beq     memwrite
000003A6 207C00500215           move.l  #(data+10),a0
000003AC 227900000006           move.l  dma,a1         * Daten in Sektorbuffer Z80 Raum schreiben
000003B2 363C007F               move.w  #127,d3
000003B6 1099           write1: move.b  (a1)+,(a0)
000003B8 5488                   addq.l  #2,a0
000003BA 51CBFFFA               dbf     d3,write1
000003BE 323900000002           move.w  track,d1       * Parameter an Z80
000003C4 343900000004           move.w  sector,d2
000003CA 163900000000           move.b  seldrv,d3
000003D0 6114                   bsr     bz80           * Befehl an Z80
000003D2 10390050020D           move.b  data+2,d0      * Bios Fehlermeldung an 68K Bdos
000003D8 4E75                   rts

                        memwrite:
000003DA 61A4                   bsr     getad
000003DC 227900000006           move.l  dma,a1
000003E2 C149                   exg     a0,a1
000003E4 608C                   bra     datamove

                        *       Schnittstelle zwischen den beiden Prozessoren

000003E6 48E700C0       bz80:   movem.l a0/a1,-(a7)
000003EA E448                   lsr.w   #2,d0          * Funktionsnummer regenerieren
000003EC 13C000500207           move.b  d0,combyte     * an Z80 Treiber übergeben
000003F2 207C0050020B           move.l  #data,a0       * Parameter kopieren
000003F8 11410002               move.b  d1,2(a0)
000003FC E049                   lsr.w   #8,d1
000003FE 1081                   move.b  d1,(a0)
00000400 11420006               move.b  d2,6(a0)
00000404 E04A                   lsr.w   #8,d2
00000406 11420004               move.b  d2,4(a0)
0000040A 11430008               move.b  d3,8(a0)
0000040E 13FC00FF00500209       move.b  #startz80,ready * Startsignal für Z80
00000416 0C3900FF00600001 bz80a: cmpi.b #readyz80,slatch       * Warten auf $ff im Systemlatch
0000041E 66F6                   bne     bz80a
00000420 227900200008           movea.l zlat,a1        * Adresse Systemlatch aus EPROM
00000426 4211                   clr.b   (a1)           * Inhalt Systemlatch löschen
00000428 4CDF0300               movem.l (a7)+,a0/a1
0000042C 4E75                   rts

                        flush:
0000042E 4280                   clr.l   d0             * o.k.
00000430 4E75                   rts

                        getseg:
```

```
464 00000432 203C0000000A              move.l  #memrgn,d0      * zurück mit Adresse der Speichereinteilungs-
465 00000438 4E75                      rts                     * tabelle
466
467                             *       Nicht installiert
468 0000043A 4240               getiob: clr.w   d0
469 0000043C 103900500007               move.b  iobyte,d0
470 00000442 4E75                       rts
471
472 00000444 13C100500007       setiob: move.b  d1,iobyte
473 0000044A 4E75                       rts
474
475                             setexc:
476 0000044C 0281000000FF               andi.l  #$ff,d1         * nur für exceptions 0 - 255
477 00000452 E549                       lsl     #2,d1           * mult. exception-Nummer mit 4
478 00000454 2041                       movea.l d1,a0
479 00000456 2010                       move.l  (a0),d0         * hole ursprünglichen Vektor
480 00000458 2082                       move.l  d2,(a0)         * Schreibe neuen Vektor
481 0000045A 4E75                       rts
482
483 00000000                            data
484 00000000 FF                 seldrv: .dc.b   $ff     * Speicher für Laufwerksnummer
485
486 00000002                            .even
486 00000002
487 00000002 0000               track:  .dc.w   0       * Speicher für Track und Sektor
488 00000004 0000               sector: .dc.w   0
489 00000006 00000000           dma:    .dc.l   0       * Speicher für DMA Adresse
490
491 0000000A 0001               memrgn: .dc.w   1       * eine Speicherverteilung
492 0000000C 00008000                   .dc.l   begtpa  * Beginn der TPA
493 00000010 00000000                   .dc.l   0       * Länge wird bei der Init. festgelegt
494 00000014 547261636B6F6666   initerr:.dc.b   'Trackoffset zu klein. System hat Ramfloppy-Directory zerstört !',0
494 0000001C 73657207A75206B
494 00000024 6C65696E2E2E205379
494 0000002C 7374656D20686174
494 00000034 2052616D666C6F70
494 0000003C 70792D4469726563
494 00000044 746F7279207A6572
494 0000004C 73747C7274202100
495 00000054 0D0A0A0A           cbootmsg:       .dc.b   $d,$a,$a,$a
496 00000058 2A2A2A2A2A2A2A2A           .dc.b   '**************************************************************'
496 00000060 2A2A2A2A2A2A2A2A
496 00000068 2A2A2A2A2A2A2A2A
496 00000070 2A2A2A2A2A2A2A2A
496 00000078 2A2A2A2A2A2A2A2A
496 00000080 2A2A2A2A2A2A2A2A
496 00000088 2A2A2A2A2A2A2A2A
496 00000090 2A2A2A2A2A
497 00000095 0D0A                       .dc.b   $d,$a
498 00000097 2A20202020202020          .dc.b   '*       S W S  -  6 8 0 0 0      BIOS Version 1.5        *'
498 0000009F 2053205720532020
498 000000A7 2D2020362036382030
498 000000AF 2030203020202020
498 000000B7 2020204249F5320
498 000000BF 5665727273696F6E20
498 000000C7 312E352020202020
498 000000CF 202020202A
499 000000D4 0D0A                       .dc.b   $d,$a
500 000000D6 2A20202020202020          .dc.b   '*       CP/M 68K                TPA  *'
500 000000DE 2043502F4D203638
500 000000E6 4B2020202020202020
500 000000EE 2020202020202020
500 000000F6 2020205450412020
500 000000FE 24
501 000000FF 2E2E2E2E2E2E202D   tpa1:   .dc.b   '..... - $'
501 00000107 2024
502 00000109 2E2E2E2E2E2E2020   tpa2:   .dc.b   '.....  $'
502 00000111 202A
503 00000113 0D0A                       .dc.b   $d,$a
504 00000115 2A2A2A2A2A2A2A2A           .dc.b   '*************************************************************'
504 0000011D 2A2A2A2A2A2A2A2A
504 00000125 2A2A2A2A2A2A2A2A
504 0000012D 2A2A2A2A2A2A2A2A
504 00000135 2A2A2A2A2A2A2A2A
504 0000013D 2A2A2A2A2A2A2A2A
504 00000145 2A2A2A2A2A2A2A2A
504 0000014D 2A2A2A2A2A
505 00000152 0D0A0A00                   .dc.b   $d,$a,$a,0
506
507
508                             *       Nicht initialisierte Speicher
509 00000000                            .bss
510                                     .even
511 00000000                    dirbuf: .ds.b   128
512 00000080                    distable:       .ds.w   16
513                             *       Adresse des ersten dph's und Reservierung von 2K freiem Speicher
514                             *       für Laufwerkstabellen und Verwaltung
515 000000A0                    dph0:   .ds.b   $800
516
517 000008A0                            end
```

Symbol Table

_ccp	********	EXT	_init	00000000	TEXT	begtpa	00008000	ABS	biosbase	00000256	TEXT
bz80	000003E6	TEXT	bz80a	00000416	TEXT	cbootmsg	00000054	DATA	combyte	00500207	ABS
conin	000002B8	TEXT	concut	000002C6	TEXT	constat	000002B8	TEXT	copydpar	000001A2	TEXT
copydpb	000001EA	TEXT	copyex	000001CC	TEXT	data	0050020B	ABS	datam1	00000376	TEXT
datamove	00000372	TEXT	dirbuf	00000000	BSS	distable	00000080	BSS	dma	00000006	DATA
dpbex	00000244	TEXT	dph0	000000A0	BSS	dphlen	0000001A	ABS	error	00000368	TEXT
flush	0000042E	TEXT	gdex	000001E6	TEXT	getad	00000380	TEXT	getdph	0000014E	TEXT
getintdr	000001D0	TEXT	getiob	0000043A	TEXT	getseg	00000432	TEXT	home	000002C6	TEXT
init1	0000000A	TEXT	init10	000000FE	TEXT	init2	00000026	TEXT	init3	00000038	TEXT
init4	00000046	TEXT	init5	0000006B	TEXT	init6	00000070	TEXT	init7	0000009E	TEXT
initerr	00000014	DATA	iobyte	00500007	ABS	liststt	000002B8	TEXT	lstout	000002C6	TEXT
memflo	00000000	ABS	memread	0000036A	TEXT	memrgn	0000000A	DATA	memwrite	000003DA	TEXT
nfuncs	00000017	ABS	outadr	0000010C	TEXT	outn	00000122	TEXT	outn1	00000132	TEXT
outnb	0000011A	TEXT	outnw	00000112	TEXT	outsl	00000148	TEXT	outslp	0000013A	TEXT
outstr	00000136	TEXT	pun	000002C6	TEXT	rdr	000002B8	TEXT	read	00000322	TEXT
readl	00000360	TEXT	ready	00500209	ABS	readyz80	000000FF	ABS	rz80byte	00000180	TEXT
rz80word	00000192	TEXT	sector	00000004	DATA	sectran	00000306	TEXT	seldrv	00000000	DATA
seldsk	000002CC	TEXT	selrtn	000002F4	TEXT	setdma	0000031A	TEXT	setexc	0000044C	TEXT
setiob	00000444	TEXT	setsec	000002FE	TEXT	settrk	000002F6	TEXT	slatch	00600001	ABS
solflo	00200000	ABS	startz80	000000FF	ABS	tpa1	000000FF	DATA	tpa2	00000109	DATA
track	00000002	DATA	traphnd1	00000246	TEXT	trapng	00000254	TEXT	tstdsk	00000164	TEXT
wboot	000002B2	TEXT	write	00000396	TEXT	write1	000003B6	TEXT	xseldsk	00000009	ABS
z80mem	00500001	ABS	zlat	00200008	ABS						

BIOS

ausgeführt, das lediglich die Funktionsnummer an das Z80-BIOS übergibt, und dessen Ergebnisse in das Register D0.W lädt. Da die oben angeführten Funktionen stets nur 8-Bit-Ergebnisse liefern, genügt es, das Datenbyte „data1" in das Register D0.B zu laden. Die Bits 8 bis 15 von D0 sind immer 0. Noch einfacher lassen sich die Funktionen Konsolenausgabe (4), Druckerausgabe (5), Lochstreifenstanzer (6) und Home (8) (Kopf auf Spur 0 stellen) realisieren. Hier werden Funktionsnummer und Datum an den Z80 übergeben und die Ausführung des Kommandos abgewartet. Die Routine zur Anwahl eines bestimmten Laufwerks (9) verwendet die Laufwerkstabelle (genaue Beschreibung siehe unten), die von der Kaltstartprozedur aufgebaut wurde, um dessen Anwesenheit festzustellen und gibt in D0.L die Adresse des zugehörigen DPHs aus. In den Unterprogrammen zur Spur- (10) und Sektorauswahl (11) sowie zur Festlegung der DMA-Adresse (12) werden die Daten lediglich bis zum physikalischen Laufwerkszugriff zwischengespeichert. Die einzigen Funktionen, die eine gewisse Sonderbehandlung verlangen, sind die Schreib-, Leseroutinen für die Laufwerke und die für die Umsetzung der logischen in physikalische Sektornummern. Die Funktion zur Wandlung der Sektornummern (16) übergibt im Register D1.W die logische Sektornummer und in D2.W die Adresse der Sektorübersetzungstabelle im Z80-Raum. Diese Adresse wurde von der Kaltstartroutine in die DPHs kopiert und kann jetzt wieder an den Z80 zur Lokalisierung der Tabelle übergeben werden. Die physikalische Sektornummer wird wie immer im Register D0.W ans BDOS geliefert.

Die RAM-Floppy, etwas Besonderes

Bei den Schreib-/Lesevorgängen muß zwischen Zugriffen auf die 68000-RAM-Floppy und die restlichen Laufwerke unterschieden werden. Dies geschieht mit Hilfe der an Adresse 0 abgelegten Laufwerksnummer der RAM-Floppy. Bei Zugriffen auf die RAM-Floppy wird zu speziellen Treibern verzweigt, in denen der 68000 die Daten direkt von der Quell- zur Zieladresse bewegt. Der Verkehr mit den übrigen Laufwerken erfolgt wieder über den Z80. Der „kleinen" CPU werden dabei gleichzeitig mit dem Aufruf der Schreib- bzw. Leseroutinen alle notwendigen Parameter übergeben. Bei Schreibzugriffen sind die Daten dann bereits im Sektorpuffer des Treibers abgelegt worden. Bei Lesezugriffen werden sie abgeholt, nachdem der Z80 den Vollzug des Befehls gemeldet hat. Damit sind sämtliche BIOS-Funktionen behandelt, zu denen der Z80 einen Beitrag leisten muß. Die restlichen Routinen betreffen nur den 68000.

Die TPA, variabel

Eine wichtige BIOS-Funktion ist Nummer 18, mit der das BDOS den Beginn und die Länge der TPA feststellen kann. Die notwendigen Daten werden von der Kaltstartinitialisierung in die Tabelle eingetragen, so daß die Größe der TPA von der Größe der installierten RAM-

DPB bei 512 KByte Speicher und 160 KByte TPA

```
;       DISKS   1
dpbase  equ     $               ;Base of Disk Parameter Blocks
dpe0:   dw      xlt0,0000h      ;Translate Table
        dw      0000h,0000h     ;Scratch Area
        dw      dirbuf,dpb0     ;Dir Buff, Parm Block
        dw      csv0,alv0       ;Check, Alloc Vectors
;       DISKDEF 0,1,64,0,2048,160,128,128,24
;
;        2560:  128 Byte Record Capacity
;         320:  Kilobyte Drive  Capacity
;         128:  32 Byte Directory Entries
;         128:  Checked Directory Entries
;         256:  Records / Extent
;          16:  Records / Block
;          64:  Sectors / Track
;          24:  Reserved  Tracks
;
dpb0    equ     $               ;Disk Parameter Block
        dw      64              ;Sectors Per Track
        db      4               ;Block Shift
        db      15              ;Block Mask
        db      1               ;Extnt Mask
        dw      159             ;Disk Size - 1
        dw      127             ;Directory Max
        db      192             ;Alloc0
        db      0               ;Alloc1
        dw      32              ;Check Size
        dw      24              ;Offset
xlt0    equ     0               ;No Translate Table
als0    equ     32              ;Allocation Vector Size for variable TPA
css0    equ     32              ;Check Vector Size
;       ENDEF
;
;       Uninitialized Scratch Memory Follows:
;
begdat  equ     $               ;Start of Scratch Area
dirbuf: ds      128             ;Directory Buffer
alv0:   ds      als0            ;Alloc Vector
csv0:   ds      css0            ;Check Vector
enddat  equ     $               ;End of Scratch Area
datsiz  equ     $-begdat        ;Size of Scratch Area
        db      0               ;Marks End of Module
;       DISKS   1
dpbase  equ     $               ;Base of Disk Parameter Blocks
dpe0:   dw      xlt0,0000h      ;Translate Table
        dw      0000h,0000h     ;Scratch Area
        dw      dirbuf,dpb0     ;Dir Buff, Parm Block
        dw      csv0,alv0       ;Check, Alloc Vectors
;       DISKDEF 0,1,64,0,2048,416,128,128,24
;
;        6656:  128 Byte Record Capacity
;         832:  Kilobyte Drive  Capacity
;         128:  32 Byte Directory Entries
;         128:  Checked Directory Entries
;         128:  Records / Extent
;          16:  Records / Block
;          64:  Sectors / Track
;          24:  Reserved  Tracks
;
dpb0    equ     $               ;Disk Parameter Block
        dw      64              ;Sectors Per Track
        db      4               ;Block Shift
        db      15              ;Block Mask
        db      0               ;Extnt Mask
        dw      415             ;Disk Size - 1
        dw      127             ;Directory Max
        db      192             ;Alloc0
        db      0               ;Alloc1
        dw      32              ;Check Size
        dw      24              ;Offset
xlt0    equ     0               ;No Translate Table
als0    equ     64              ;Allocation Vector Size for variable TPA
css0    equ     32              ;Check Vector Size
;       ENDEF
;
;       Uninitialized Scratch Memory Follows:
;
begdat  equ     $               ;Start of Scratch Area
dirbuf: ds      128             ;Directory Buffer
alv0:   ds      als0            ;Alloc Vector
csv0:   ds      css0            ;Check Vector
enddat  equ     $               ;End of Scratch Area
datsiz  equ     $-begdat        ;Size of Scratch Area
        db      0               ;Marks End of Module
```

Bild 2. Zwei DPBs für die RAM-Floppy

Floppy abhängt. Damit hat der Anwender eine einfache Möglichkeit, den Speicher zwischen Betriebssystem und RAM-Floppy aufzuteilen. Welcher Parameter dabei den entscheidenden Einfluß auf die Speicherplatzverteilung hat, das wird später noch besprochen. Die BIOS-Funktionen 19 und 20 bedienen direkt das IO-Byte von CP/M 2.2 an der Adresse 3 des Z80-Adreßraums. Dadurch ist es auch mit CP/M-68k möglich, verschiedene Ein-/Ausgabekanäle anzuwählen und sämtliche Möglichkeiten auszunutzen, die das 8- Bit-CP/M bietet. Die Funktion 21, die das Leeren aller Laufwerkspufferspeicher bewirkt, ist hier nicht installiert, da diese Aufgabe vom 8-Bit-System selbstständig erfüllt werden muß. Die letzte Funktion (22) ermöglicht die Veränderung der Trapvektoren 0 ... 255 für die 68000-Interruptverarbeitung.

Die Speicheraufteilung wählbar

Als nächstes muß die Frage nach einer sinnvollen Speicheraufteilung gestellt werden. Da es unter CP/M-68k fast keine Einschränkungen in Bezug auf die Speicherverteilung gibt, wurde hier die folgende Wahl getroffen.

$000000 bis $0003FF Bereich für die Interruptvektoren des 68000 und Stackbereich für die RAM-Floppytreiber.

$000400 bis $0015FF RAM-Bereich für Monitorvariable und Monitorerweiterungen.

$001600 bis $007FFF Betriebssystem
$008000 bis $XXXXXX TPA . Dabei errechnet sich die Größe XXXXXX wie folgt: XXXXXX = $2000 * Track Offset der RAM-Floppy. Die aktuelle Größe der TPA wird nach jedem Systemeintritt in der Kaltstartmeldung ausgegeben.

$XXXXXX bis $ZZZZZZ Speicher für RAM-Floppy ($ZZZZZZ = RAM TOP).

Um über die Größe der RAM-Floppy bzw. der TPA wirklich frei verfügen zu können, wurde das Betriebssystem in den unteren Adreßbereich des 68000 gelegt. Dadurch kann die Grenze zwischen TPA und RAM-Floppy beliebig verschoben werden ohne daß das Betriebssystem seine Lage andern müßte. Nach jedem Kaltstart des CP/M 2.2 müssen die Parameter im DPB (Zahl der Blöcke und der reservierten Spuren) der RAM-Floppy entsprechend eingesetzt werden.

Mit dem Turbo-Pascal-Programm in *Bild* 3 kann man die Laufwerksparameter einstellen. Es initialisiert gleichzeitig die RAM-Floppy. Durch die Cursor-Steuertasten (Crtl-X <down> und Crtl-E <up> – wie bei WordStar) kann die Größe der TPA in 8-KByte-Schritten verändert werden. Man kann die Untergrenze für die RAM-Floppy soweit herabsetzen, daß nur noch die ersten 8 KByte des Speichers für den Monitor und den 68000-Stack übrig bleiben. Dies ist aber nur sinnvoll, wenn man kein 16-Bit-Betriebssystem laden will. Sobald das System nicht mehr geladen werden kann (negative Werte für die TPA Größe) erscheint eine entsprechende Meldung.

Wer es trotzdem versucht, zerstört den Inhalt der RAM-Floppy-Directory und damit alle Daten. Eine Veränderung der RAM-Floppyuntergrenze führt ebenfalls zum Verlust aller Daten, da die Directory-Informationen immer in den ersten Sektoren der ersten erlaubten Spur stehen. Daraus folgt, daß man sich am besten zu Anfang genau überlegt, wie groß die TPA oder die RAM-Floppy im Normalbetrieb sein soll, und anschließend an dieser Konfiguration nichts mehr ändert. Jeder Kaltstart des CP/M 2.2 zieht eine neuerliche Festlegung der RAM-Floppygröße nach sich, falls diese mit

```
Program InitRamfloppy;

const
     ramstep = 8;( Ramfloppygroesse kann nur in 8 Kbyte Schritten veraendert
                   werden)
     miniram = 8;( Minimale Ramgroesse 8 KByte )
     seldsk  = 8;( Nummer der Bios-Funktion ( - 1 ) zur Auswahl eines Laufwerks.
                   Liefert in HL die Adresse des zugehoerigen DPH's )
     systemsize = 32 ; ( System, Monitorspeicher und Stack belegen 32 KByte )
var
   ramsiz,tpasiz,dpbbase,dphbase,blocksiz : integer;
   dpb : array[0..14] of byte;( Kopie des Ramfloppy DPB's)
   system : string[4];
   buffer : array[0..127] of byte;
   parfield  : Integer absolute $2000;
   ramfloppy : byte absolute $2002; ( Patchbereich um Ramfloppynummer und )
   latch68   : byte absolute $2003; ( Adresse des 68000 Latchs einzustellen )
                                    ( Voreinstellung : parfield  = $10
                                                       Ramfloppy = 1
                                                       latch68   = $ff )

Procedure ReadDPB;

var
   i : integer ;
begin
     dphbase:=BiosHL(seldsk,ramfloppy);( DPH Adresse der Ramfloppy feststellen )
     dpbbase:=mem[dphbase+10]+(mem[dphbase+11] shl 8);( DPB Adresse holen )
     for i:=0 to 14 do dpb[i]:=mem[i+dpbbase];( DPB auslesen )
     blocksiz:=1; for i:= 1 to dpb[2] do
                  blocksiz:=blocksiz*2;( Blockgroesse in 128 Byte Sektoren )
     ramsiz:= ((dpb[5]+1+dpb[6]*$100)*blocksiz) div 8; ( Ramgroesse in KByte )
     tpasiz:= (dpb[13]+dpb[14]*$100)*8-systemsize; ( Groesse der TPA )
end;

Procedure SetPar;

var i : integer;

begin
     dpb[5]:= ((ramsiz*8)div blocksiz - 1) mod 256;
     dpb[6]:= ((ramsiz*8)div blocksiz - 1) div 256;
     dpb[13]:=((tpasiz+systemsize) div 8 ) mod 256;
     dpb[14]:=((tpasiz+systemsize) div 8 ) div 256;
     for i:=0 to 14 do mem[dpbbase+i]:=dpb[i];
end;

Procedure Ready;

begin
     repeat until mem[parfield]=$ff;
     mem[parfield]:=0;
end;

Procedure Initfloppy;
var
   maxtrack,track,sector,dmaadr,i: integer;

procedure SetParfield;
begin
          mem[parfield]:=0;
```

Bild 3. Pascalprogramm zur Initialisierung der RAM-Floppy und Speicheraufteilung

```
        mem[parfield+1]:=addr(buffer) mod 256;
        mem[parfield+2]:=addr(buffer) div 256;
        mem[parfield+3]:=track;
end;

begin
     for i:= 0 to 127 do buffer[i]:=$E5;
     maxtrack:= (ramsiz+tpasiz+systemsize) div 8 - 1;
     for track:= (tpasiz+systemsize) div 8 to maxtrack do
     begin
          setparfield;
          for sector := 0 to 63 do begin
              gotoxy(18,23);write(track:4); gotoxy(37,23);write(sector:4);
              mem[parfield+4]:=sector;
              port[latch68]:=2;
              ready;
          end;
     end;
     buffer[0]:=ramfloppy;{ Ramfloppy Laufwerksnummer an 68000 Adresse 0 schreiben !}
     track:=0;
     setparfield;
     mem[parfield+4]:= 0;
     port[latch68]:=2;
     ready;
end;

Procedure Screen;

begin
     gotoxy(10,1); write('*********************************************');
     gotoxy(10,2); write('* --->> Aufteilung des 68000 Speichers <<--- *');
     gotoxy(10,3); write('*********************************************');
     gotoxy(10,5); write('Groesse der CP/M 68k TPA    :           KByte');
     gotoxy(10,7); write('Groesse der Ramfloppy       :           KByte');
     gotoxy(10,9); Write('System ladbar ?             :');
     gotoxy(10,12);Write('mehr TPA : Cursor up ( Crtl-E ) , ');
     write ('weniger TPA : Cursor down ( Crtl-X )');
     gotoxy(10,14);Write('Ende     : E');
end;

Procedure Show;

begin
     gotoxy(43,5); write(tpasiz:5);
     gotoxy(43,7); write(ramsiz:5);
     gotoxy(44,9); write(system);
end;

{---------------------------- Hauptprogramm ----------------------------}
var
   key : char;
begin
clrscr;
readdpb;
if tpasiz>0 then system := '  Ja' else system := 'Nein';
screen;
show;
repeat
      read(kbd,key);key:=upcase(key);
      if key = ^E then begin
         tpasiz:=tpasiz+ramstep; ramsiz:=ramsiz-ramstep; end;
      if key = ^X then begin
         tpasiz:=tpasiz-ramstep; ramsiz:=ramsiz+ramstep; end;
      if tpasiz < -24 then begin
         tpasiz:=tpasiz+ramstep; ramsiz:=ramsiz-ramstep; end;
      if ramsiz <=0 then begin
         tpasiz:=tpasiz-ramstep; ramsiz:=ramsiz+ramstep; end;
      if tpasiz>0 then system := '  Ja' else system := 'Nein';
      show;
until key = 'E';
gotoxy(10,16); write('Sollen diese Parameter eingesetzt werden ( J/N ) ?  : ');
read(kbd,key);key:=upcase(key);write(key);
if key = 'J' then begin setpar;
   gotoxy(10,18); write('------->> DPB aktualisiert <<--------');
   gotoxy(10,20); write('Soll die Ramfloppy initialisiert werden (J/N) ? : ');
   read(kbd,key);key:=upcase(key);write(key);
   if key = 'J' then begin
      gotoxy(10,22);write('Ramfloppy wird initialisiert !!!');
      gotoxy(10,23);write('Track :          Sektor :');initfloppy;end;
end;
clrscr;
end.
```

dem Programm von oben vorher eingestellt wurde. Deshalb sollte man sich bei der Anpassung des 8-Bit-BIOS für eine Aufteilung des 68000 Speichers entscheiden, die den meisten Anforderungen genügt. Dann findet man auch nach einem Kaltstart die Directory der RAM-Floppy wieder, ohne erst lange überlegen zu müssen, wie groß die RAM-Floppy vor dem Kaltstart war. Zur Installation des Turbo-Pascal-Programmes sollte noch gesagt werden, daß die Variablen „parfield", „RAM-Floppy" und „latch68" von Lesern, bei denen die hier veröffentlichte Voreinstellung nicht stimmt und die keinen Pascal-Compiler besitzen, mit einen Debugger entsprechend „gepatcht" werden müssen. Wer selbst einen Turbo-Pascal-Compiler besitzt, kann natürlich auf die absoluten Variablen verzichten und den Quelltext nach dem Ersetzen der Variablen durch Konstante nochmals übersetzen.

Für den Betrieb der RAM-Floppy mit variabler Speichergröße muß ihr DPB besondere Anforderungen erfüllen. Da die Größe der Blockbelegungstabelle (Allocation Vector) von der Zahl der zur Verfügung stehenden Blöcke abhängt, muß soviel Platz für die Tabelle reserviert werden, wie bei maximaler RAM-Floppygröße notwendig ist. Die Zahl X der zu reservierenden Bytes berechnet sich folgendermaßen:

$$x = \frac{\text{Speichergröße in KByte}}{(\text{Blockgröße in KByte}) \times 8}$$

In Bild 2 sind nochmals zwei DPBs mit ausführlichen Kommentaren dargestellt. Der erste für 512 KByte Speicher, der zweite für 1 MByte. Die TPA ist in beiden Fällen 160 KByte groß. Bei 512 KByte verbleiben 320 KByte Speicher für die RAM-Floppy, bei 1 MByte sind es 832 KByte. Mit dieser Einteilung können auch die etwas größeren Dateien des CP/M-68k problemlos geladen werden und die RAM-Floppy bietet eine ausreichende Kapazität für Source-Dateien, die dann sehr schnell vom Assembler oder Compiler übersetzt werden können.

Der Ladevorgang

Nachdem die Verteilung des Speichers festgelegt ist, folgt nun der eigentliche Ladevorgang des Betriebssystems. Dieser gestaltet sich sehr einfach, da der Z80 die notwendigen Programme von der Diskette in seinen Speicher laden kann und sie von dort direkt durch die Laderoutine des RAM-Floppytreibers in den 68000 Speicher transportiert wer-

den. Die sonst üblichen Prozeduren mit Urladeprogrammen, die ihrerseits erst das Betriebssystem laden müssen, entfallen vollständig. Das gesamte Betriebssystem mit der benötigten Z80-Software befindet sich in einer Datei mit dem Namen CPM68.COM auf der Diskette. Das Source Listing des kleinen Z80-Laderprogrammes ist in *Bild 4* dargestellt.

Der Z80 vermittelt lediglich die Ziel- und Quelladresse, sowie die Länge des Programmes über das Parameterfeld bei Adresse 10h an den 68000 und gibt dann den Ladebefehl an dessen Befehlslatch aus. Den Rest erledigt der 68000 alleine.

Sobald das System einmal geladen ist bleibt es resident im Speicher und kann beliebig oft betreten und verlassen werden. Zum Eintritt in das System wird ganz analog zum Monitor das Programm „ENTER.COM" aufgerufen. Nur muß jetzt, wie oben schon erwähnt, die Einsprungadresse die am Label „-floc" steht, auf den Beginn des CP/M-68k ($1600) gerichtet werden. Sobald die Kaltstartinitialisierung des Betriebssystems fehlerlos abgelaufen ist, meldet es sich mit seiner Kaltstartmeldung, der die aktuelle Größe der TPA entnommen werden kann, und dem üblichen CP/M Prompt 'A>'. Wird das System nicht zum erstenmal betreten, sollte nach jedem Eintritt ein Warmstart mit Crtl-C durchgeführt werden, um alle Systemvariablen zu initialisieren. Andernfalls kann es zu einem Softwareinterrupt kommen, der aber erfahrungsgemäß keine weiteren Folgen hat. Danach ist das System komplett betriebsbereit. Für den Benutzer steht jetzt ein vollständiges 68000-System zur Verfügung, dessen Ein- und Ausgaben über den Z80 laufen.

Wie bereits im ersten Teil dieser Artikelserie geschildert wurde, ist es aber nicht unbedingt notwendig, jeden Datenaustausch mit der Peripherie über den Z80 laufen zu lassen. Wer will, kann auch direkt auf die vorhandene Hardware zugreifen, ohne mit dem Z80 in Schwierigkeiten zu kommen, da dieser ja brav in der Treiberroutine seine Kreise zieht und auf Befehle von „oben" wartet. Ja es ist sogar möglich, den Z80 (4 MHz) Daten von der Diskette (bei einer Übertragungsrate von 250 KBit/s) lesen zu lassen, während der 68000 auf dessen Speicher mit RAM-Floppy-Geschwindigkeit (160 KByte/s) zugreift. Der Autor hat dies selbst ausprobiert und konnte bei Lesevorgängen keine Fehler feststellen. Der Grund, weshalb solche schon unheimlich anmutenden Übungen funktionieren, liegt darin, daß ein DMA-Zugriff des 68000 zwischen 1 und 1,25 µs dauert und im Mittel alle 6,25 µs erfolgt. Für den Z80 bedeutet das eine Verringerung seiner effektive Taktgeschwindigkeit um etwa 25% auf ca. 3 MHz. Das ist immer noch schnell genug, um alle Daten rechtzeitig vom Floppy Controller abzuholen.

Diese Zahlen zeigen, daß zwar keine vollständige Transparenz vorliegt, die Reduzierung der Z80-Leistung durch die Zugriffe des 68000 aber in einem durchaus tragbaren Rahmen liegen. Hier stehen Tüftlern und Leistungfanatikern sämtliche Möglichkeiten offen, aus ihren mittlerweile vielleicht doch schon etwas angegrauten ECB-Systemen wieder Maschinen zu machen, die sich sehen lassen können. Ein weiterer Aspekt dieser Host-Prozessorphilosophie sei hier der Vollständigkeit halber noch angesprochen. Es besteht natürlich fast keine Begrenzung die Anzahl der vorhandenen 68000-Karten betreffend (außer vielleicht durch die Zahl der Busplätze, der Leistungsfähigkeit der Stromversorgung oder auch des Geldbeutels). Da jede Karte ihren eigenen Adreßraum hat und eine eigene Adresse für ihr Befehlslatch bekommen kann, ist ein paralleles Arbeiten einer großen Zahl von Prozessoren möglich. Wer die damit verbundenen Probleme bei der Programmierung löst, verfügt über eine Maschine, die mit ihren Leistungen bestimmt in erstaunliche Regionen vorzudringen vermag. Diese Ausblicke sollten nur noch einmal auf die Fähigkeiten des mc-68-ECB aufmerksam machen, der viel mehr sein kann, als nur ein Entwicklungssystem für 68000 Software.

Die Rückkehr zum 8-Bit-System

Nach diesem Ausflug in die phantastische Welt der Multiprozessormaschinen soll jetzt das Programm besprochen werden, mit dem man wieder in das 8-Bit-Betriebssystem zurückkehren kann. Der Rückweg gestaltet sich genauso einfach, wie der Aufruf des CP/M-68k. Das in *Bild 5* dargestellte Programm gibt eine Meldung aus, wenn das 16-Bit-System verlassen wird, und veranlaßt den Z80 dazu, einen Warmstart auszuführen. Der 68000 springt wieder in die RAM-Floppytreiber. Damit ist der ursprüng-

```
                        0001  ;*************************************************************
                        0002  ;*              Lader fuer 68k CP/M                          *
                        0003  ;*************************************************************
                        0004
      (0010)            0005  parfield      equ     10h
      (0003)            0006  load          equ     3
      (00FF)            0007  syslat        equ     0ffh
                        0008
      (0100)            0009          org     100h
0100  C30B01      R     0010          jp      start
                        0011
0103  00                0012  ready:  db      0
0104  1C02              0013  source: dw      21ch            ; Beginn des 68000 Systemfiles ohne die
                        0014                                  ; Informationen für den CP/M 68k CCP
0106  001600            0015  dest68: db      0,16h,0         ; Zieladresse 68000 $1600
0109  0055              0016  length: dw      5500h           ; Länge des Systems ohne bss
      (0008)            0017  ldirlen equ     $-ready
                        0018
010B  210301            0019  start:  ld      hl,ready
010E  111000            0020          ld      de,parfield
0111  010800            0021          ld      bc,ldirlen
0114  EDB0              0022          ldir                    ; Parameterfeld initialisieren
0116  3E03              0023          ld      a,load
0118  D3FF              0024          out     syslat,a        ; Befehl abschicken
                        0025
011A  3A1000            0026  loop:   ld      a,(parfield)
011D  B7                0027          or      a
011E  28FA              0028          jr      z,loop          ; Auf Ausführung warten
0120  AF                0029          xor     a
0121  321000            0030          ld      (parfield),a
0124  C9                0031          ret
                        0032
      (0200)            0033          org     200h
                        0034
0200  (5500)            0035          ds      5500h           ; Platz für das 68k System
                        0036
5700  (0000)            0037          end

Errors                   0
```

Bild 4. Der Lader für CP/M-68k

```
*********************************************************
*       Verlassen des 68k CPM                           *
*********************************************************

z80mem  equ     $500001         * Beginn des Z80 Adressraums
combyte equ     2+$103+z80mem   * Kommandobyte
ready   equ     combyte+2       * Fertig 68000 --> Z80

quit    equ     $ff             * Kommando für Warmstart des Z80
startz80        equ     $ff     * Veranlaßt den Z80 zum Verlassen der
*                               Warteschleife
zlatch  equ     $4001ff         * Systemlatch vom 68000 aus gesehen
solflo  equ     $20000c         * Einsprung in die Ramfloppy Treiber

*       68k CP/M BDOS Funktionen
outstr  equ     9


start:  move.w  #outstr,d0
        move.l  #msg,d1         * Ausgabe des 'Ende'-Strings
        trap    #2              * BDOS Aufruf
        move.b  #quit,combyte   * Kommando an Z80
        move.b  #startz80,ready * Start  Z80
        clr.b   zlatch          * Löschen des Systemlatches
        jmp     solflo

msg:    dc.b    $d,$a,'--->> Sie verlassen das 68k CP/M ! Warmstart ausführen !'
        dc.b    $d,$a,'$'
        end
```

Bild 5. Zurück zum 8-Bit-System

liche Zustand hergestellt. Wird der Warmstart des CP/M 2.2 zum Beispiel durch MicroShell abgefangen, so sollte er per Hand ausgeführt werden. Der Grund dafür wurde schon einmal bei der Besprechung der RAM-Floppy-Installation angesprochen. Der 68000 kann den Inhalt sämtlicher Massenspeicher des Systems ohne das Wissen des CP/M 2.2 verändern. Nach einer Rückkehr ins 8-Bit-System kann der Schreibversuch eines Programms zu der gefürchteten Fehlermeldung 'BDOS ERR ON X' führen und die mit diesem Programm erarbeiteten Daten sind verloren. Deshalb sollte man nach jedem Wechsel der Betriebssysteme einen Warmstart ausführen.

Der Kaltstart

Zum Abschluß dieses Artikels wird der Programmablauf innerhalb der Kaltstartroutine des 68k-BIOS genau erläutert.

Die erste Aufgabe ist die Einrichtung der notwendigen DPHs und DPBs zur Verwaltung der Laufwerke. Zum Aufbau von DPHs und DPBs wird eine Tabelle eingerichtet, in der alle vorhandenen Laufwerke vermerkt werden. Die Anwesenheit der Laufwerke kann durch den Aufruf der 8-Bit-BIOS-Funktion (9) festgestellt werden. Diese liefert bei Vorhandensein des Laufwerks die Adresse des entsprechenden DPHs im Register HL. Gibt es das Laufwerk nicht, so ist HL = 0 . Anhand dieser Information wird die Laufwerkstabelle aufgebaut. Die Tabelle ist 16 Worte lang. Das erste Byte jedes Wortes enthält vor der Anwesenheitskontrolle der Laufwerke $ff und jedes zweite 0. Ist ein Laufwerk vorhanden, so wird das erste Byte auf null gesetzt und im zweiten eine BIOS-interne Nummer für das Laufwerk vergeben.

Gibt es z. B. in einem System die Laufwerke A, B, M, so werden ihnen BIOS-intern die Nummern 0, 1, 2 zugeordnet, d.h. die Nummern der Laufwerke sind Zahlen in aufsteigender Reihenfolge.

Zunächst müssen die zugehörigen DPHs für jedes vorhandene Laufwerk eingerichtet werden. Dazu wird zuerst die Adresse der Sektorumrechnungstabelle im Z80-Raum in den DPH kopiert. Darauf folgen 3 Worte, die für BDOS-interne Berechnungen reserviert sind; anschließend wird die 68000-Adresse des Directory-Puffers eingetragen. Danach wird die Adresse des DPBs im Z80-Raum in den DPH kopiert. Sie wird später dazu verwendet werden, den zugehörigen DPB aus dem Z80-Raum in den 68000-Raum zu kopieren. Anschließend kommen nochmals zwei 4 Byte lange Adressen der Speicherbereiche für die Prüfsummen und die Belegungstabelle.

Diese sind aber jetzt noch nicht bekannt und werden zu einem späteren Zeitpunkt eingesetzt. Auf diese Art und Weise wird für jedes vorhandene Laufwerk eine interne Nummer vergeben und ein DPH eingerichtet. Sind alle Laufwerke durchprobiert worden, so beginnt der Aufbau der DPBs, die gleich im Anschluß an alle DPHs im Speicher stehen.

Jetzt wird einfach in der Laufwerkstabelle nachgeschaut, ob ein bestimmtes Laufwerk vorhanden ist. Ist dies der Fall, so werden die notwendigen Daten aus den DPBs im Z80-Raum in die DPBs im 68000-Raum kopiert und die Startadresse des DPBs in den zugehörigen DPH eingetragen. Im Anschluß an jeden DPB werden sofort Speicherplätze für die Prüfsummen- und Belegungstabelle reserviert und die jeweiligen Adressen im DPH ein getragen. Der Aufbau des DPBs und die Reservierung des Speicherplatzes wird für alle vorhandenen Laufwerke durchgeführt. Danach sind sämtliche Informationen, die das 68k-BDOS über die Laufwerke benötigt, aus dem Z80-Raum kopiert und das CP/M-68k ist jetzt in der Lage, alle notwendigen Berechnungen zur Verwaltung der Laufwerke vorzunehmen und dem 8-Bit-BIOS die entsprechenden Parameter zu liefern. Anschließend wird aus den Laufwerksdaten der RAM-Floppy die Größe der TPA berechnet und aktuell in die Tabelle eingesetzt, aus der das BDOS seine Informationen über die Speicherplatzverteilung beziehen kann. Zusätzlich wird die Adresse des letzten Bytes, das zur TPA gehört, in die Kaltstartmeldung eingefügt. Damit ist das BIOS komplett und kann alle geforderten Funktionen erfüllen. Die oben beschriebene Einrichtung der DPHs und DPBs sowie der Tabelle zur Speicherplatzverwaltung ist vielleicht etwas kompliziert, aber dadurch ist das 68k-BIOS völlig Hardwareunabhängig. Mit diesem Teil der Artikelserie ist die Anpassung des mc-68-ECB an den mc-CP/M-Computer und die Installation des CP/M-68k vollständig beschrieben und vorläufig abgeschlossen.

Es gibt bestimmt noch vielfältige Möglichkeiten die Fähigkeiten des Systems zu erweitern und zu verbessern, so daß die Phantasie und der Ideenreichtum der zukünftigen Benutzer gefordert sind, auf die weitere Entwicklung dieses Konzepts Einfluß zu nehmen. Das komplette Betriebssystem, inklusive aller Source-Dateien die zum Aufbau des Systems notwendig sind, befinden sich auf den Systemdisketten, die vom Franzis-Software-Service zusammen mit den Handbüchern für CP/M-68k bestellt werden können. Leiterplatten, Bausätze und auch fertige mc-68-ECB Rechnerkarten sind beim Autor des Artikels erhältlich.

Jürgen Plate

Sichere Übertragung

Datenübertragung per Funk

Die gesicherte Übertragung von Dateien zwischen zwei Computern war schon oft Thema von mc-Artikeln. Sind die beiden Computer über Leitung oder Modem miteinander verbunden, gibt es keine Probleme – passende Programme mit gesichertem Übertragungsprotokoll gibt es en masse (z. B. Kermit). Bei der Datenübertragung per Funk liegt das Problem etwas anders.

Hier geht es um die Datenübertragung per Funk, genauer per Rundfunk. Und da gibt es kein gegenseitiges Handshake, das die Wiederholung fehlerhafter Blökke erlaubt, sondern nur eine Einweg-Kommunikation. Der Sender strahlt die Daten (Texte oder Programme) ein- bis zweimal aus, womit von seiner Seite die Sache erledigt ist. Beim Empfänger sieht das anders aus. Zunächst wird hier die empfangene Sendung nach den im Amateurfunk oder bei der Modem-Übertragung bekannten Verfahren von Tonfrequenzen wieder in Computerbits umgeformt und das Ganze auf Diskette gespeichert. Unter Umständen erfolgt die Prozedur noch auf dem Umweg über die Tonbandspeicherung. Das Problem liegt nun darin, die Daten in einer Form zu übermitteln, die das Erkennen und Korrigieren von Fehlern erlaubt. Dabei muß die Fehlererkennung auf jeden Fall durch den Computer erfolgen, für die Fehlerkorrektur läßt sich auch die (natürliche) Intelligenz des Benutzers einsetzen. Bei der Softwareentwicklung wird davon ausgegangen, daß der Sender eine Datei mit ASCII-Zeichen – genau wie beim Betrieb mit Akustikkoppler – zur Sendeeinrichtung schickt und die empfangenen Daten wieder in einer Datei abgelegt werden.

Intel-Hex

Nach längerem Brüten erinnerte ich mich an ein Datenformat, das zu den Anfangszeiten der Computerei entwikkelt wurde und zu einem gewissen Standard wurde, das Intel-Hexformat. Bei diesem Format werden die einzelnen Datenbytes als zweistellige Hexadezimalzahl gespeichert. Um eine gewisse Zahl von Bytes wird dann noch ein Rahmen aus Kopfinformation und Prüfsumme gelegt. Dabei ist es völlig egal, ob es sich bei den Ausgangsdaten um Texte oder Maschinencode handelt. Die so gewonnene ASCII-Datei enthält nur noch die Ziffern von 0 bis 9, die Buchstaben von A bis F, den Doppelpunkt sowie die Zeichen für Carriage-Return und Line-Feed. Alle anderen Zeichen sind verboten. Bei nur 19 gültigen Zeichen ist es schon recht einfach, falsch übertragene Zeichen zu erkennen. Zusätzliche Sicherheit bringt die Prüfsumme. Sehen wir uns das Intel-Hexformat nun genauer an.

Alle Zeichen sind hexadezimal codiert, d. h. ein Byte wird in Form zweier Hexadezimalziffern dargestellt. Die Datei wird in Sätze fester oder variabler Länge zerlegt, wobei jeder Satz den folgenden Aufbau besitzt:

– Das erste Zeichen, ein Doppelpunkt, markiert den Anfang eines Satzes und dient der Synchronisation (falls im vorhergehenden Satz Zeichen verlorengegangen sind).
– Die nächsten beiden Zeichen enthalten die Satzlänge, d. h. die Anzahl der Bytes, die in diesem Satz übertragen werden. Bei meinem Sende-Programm ist dieser Wert fest auf $20 = 32 Bytes festgelegt.
– Danach folgt vierstellig die Startadresse für das erste Byte des Satzes. Anhand dieser Adresse läßt sich überprüfen, ob Sätze verlorengegangen sind.
– Darauf folgt zweistellig die Satzart, die in unserer Anwendung immer $00 ist.
– Nun kommen der Längenangabe entsprechend viele zweistellige Hexadezimalzahlen, bei uns für 32 Bytes 64 Ziffern.
– Als letzter Wert kommt eine zweistellige Prüfsumme. Zum Berechnen der Prüfsumme werden zunächst alle Werte, beginnend bei der Längenangabe, addiert und dann das Komplement vom Low-Byte des Ergebnisses genommen.
– Der Satz wird mit Carriage-Return und Line-Feed abgeschlossen.

Aus Benutzersicht ist die Sendedatei zunächst einmal eine Textdatei, deren Zeilen (bei 32 Bytes je Satz) genau 74 Zeichen (ohne CR-LF) lang sind. So eine Textdatei läßt sich mit einem beliebigen Editor ändern und korrigieren – falls das Emfangsprogramm Fehler gefunden hat. Wenn die Hex-Datei dann noch zweimal übertragen wird, ist die Wahrscheinlichkeit recht groß, daß aus den beiden empfangenen Dateien eine vollständig richtige „zusammenkopiert" werden kann.

Wir drucken hier beide Turbo-Pascal-Programme, GENHEX und READHEX ab, da sich das Verfahren natürlich auch für ganz normale DFÜ- Anwendungen eignet.

GENHEX

Das Programm (*Bild 1*) generiert aus einer beliebigen Datei eine Datei im Intel-Hexformat, die dann übertragen werden kann. Die Quelldatei kann auch ein Maschinenprogramm enthalten. Aus diesem Grund wird sie mit der Blockread-Prozedur gelesen. Blockread legt immer ein Vielfaches von 128 Bytes in einem Puffer ab, der im Programm 8 KByte groß ist. Die Prozedur Make_Hex_Lines wandelt einen 128-Byte Block zu vier Sätzen im Hexformat um. Der letzte Satz wird gegebenenfalls auf die Länge 32 aufgefüllt. Zum Schluß wird noch ein Dateiende-Satz mit der Länge 0 geschrieben. Der Aufruf des Programms zum Konvertieren von „Eingabedatei" in eine „Ausgabedatei" im Hexformat lautet „GENHEX Eingabedatei Ausgabedatei". Bei falscher Parameterzahl verabschiedet sich das Programm mit einer Fehlermeldung. Während des Programmlaufs wird der Fortschritt der Konvertierung protokolliert.

READHEX

Wegen der Fehlerprüfung ist das Programm (*Bild 2*) wesentlich umfangreicher. Der Aufruf des Programms erfolgt

```
Program Genhex;
{$u-,c-,r+,v-}
{-------------------------------------------------------------}
{                                                             }
{  Generiert eine Intel-Hex-Datei aus einer beliebigen        }
{  Eingabe-Datei.                                             }
{                                                             }
{  Programmaufruf:                                            }
{  GENHEX <Eingabedatei> <Ausgabedatei>                       }
{                                                             }
{  <Ausgabedatei> = "LST:" ---> Drucker                       }
{  <Ausgabedatei> = "CON:" ---> Bildschirm                    }
{                                                             }
{  Jürgen Plate 5.1.1987                                      }
{                                                             }
{-------------------------------------------------------------}
const
  MaxBuffer = 8191;

type
  HexZahl    = String[4];
  Zeile      = String[80];
  InBuffer   = Array [0..MaxBuffer] of Byte;

var
 Infile     : File;       { Eingabedatei (*.COM) }
 Outfile    : Text;       { Ausgabedatei (*.HEX) }
 Puffer     : InBuffer;   { Eingabepuffer        }
 InfileSize : Integer;    { Größe der Eingabedat.}
 Max, Start : Integer;    { Blockzähler für Eing.}
 Count, I, J: Integer;    { Hilfvariablen        }
 X, Y, Line : Integer;    { Anzeigesteuerung     }

Function Hex_Byte(X : Integer): HexZahl;
  { Wandelt Byte in 2-stellige Hexzahl }
  const
    HexZiffer : Array [0..15] of Char = '0123456789ABCDEF';
  begin
  Hex_Byte := HexZiffer[Lo(x) shr 4]+HexZiffer[Lo(X) and $F];
  end;

Function Hex_Word(X : Integer): HexZahl;
  { Wandelt Integer in 4-stellige Hexzahl }
  begin
  Hex_Word := Hex_Byte(Hi(x))+Hex_Byte(Lo(x));
  end;

Procedure Make_Hex_Lines(Num : Integer);
  { erzeugt 4 Zeilen im Intel-Hexformat, je Zeile 32 Bytes = 128 Bytes }
  var
    i, j, b : Integer;
    Checksum   : Integer;
    Hilf : Zeile;
  begin
  for i:=0 to 3 do
    begin
    { Länge (immer $20), Adresse, Satzart (immer 00) }
    Hilf := '20'+Hex_Word($80+((Num*128)+(i*32)))+'00';
    Checksum := 32+((Num+1) div 2)+(i*32)+((Num+1) and 1)*128;
    { 32 Bytes in der Zeile generieren }
    for j:=0 to 31 do
      begin
      b := Puffer[((Count-Start-1) shl 7)+(i shl 5)+j];
      Hilf := Hilf+Hex_Byte(b);
      Checksum := CHecksum+b;
      end;

    Hilf := ':'+Hilf+Hex_Byte(256-Lo(Checksum));
    Writeln(Outfile,Hilf);
    GotoXY(X,Y); Line := Line+1; Write(Line:6);
    end;
  end;

begin
LowVideo;
clrscr;
Writeln;
Writeln('═══════════════════════════════════════════════════════');
Writeln('GENHEX - Erzeugen von Dateien im Intel-Hexformat');
Writeln('       Copyright (c) Franzis-Verlag 1987       ');
Writeln('═══════════════════════════════════════════════════════');
{ Kommandoparameter bearbeiten }
if ParamCount<>2 then
  begin
  Writeln('Parameterfehler: GENHEX INFILE OUTFILE');
  halt;
  end;
{$I-}
assign(Infile,Paramstr(1)); reset(Infile);
if IOResult<>0 then
  begin
  Writeln('Eingabedatei nicht vorhanden!');
  halt;
  end;
assign(Outfile,Paramstr(2)); Rewrite(Outfile);
if IOResult<>0 then
  begin
  Writeln('Ausgabedatei kann nicht geöffnet werden!');
  halt;
  end;

{ Datei konvertieren, der letzte Satz wird auf Länge $20 aufgefüllt }
InfileSize := FileSize(Infile); Line := 0;
Writeln('Generiere "',Paramstr(2),'" aus "',Paramstr(1),'"');
Writeln('Konvertiere ',InfileSize*128,' Bytes');
Write('Zeile Nummer: '); X := WhereX; Y := WhereY; Writeln(Line:6);
Writeln('═══════════════════════════════════════════════════════');
Max := (MaxBuffer+1) div 128; Start := 0;
while Start<InfileSize do
  begin
  if InfileSize-Start<Max then Max := InfileSize-Start;
  BlockRead(Infile,Puffer,Max);
  for Count := Start+1 to Start+Max do
    Make_Hex_Lines(Count);
  Start := Start+Max;
  end;
{ Abschlußzeile mit Länge 0 schreiben }
Writeln(Outfile,':0000000000');
Close(Infile);
Close(Outfile);
Writeln; Writeln;
end.
```

Bild 1. Das Programm zum Generieren einer Hex-Datei

```
Program Readhex;
{$u-,c-,r+,v-}
{-------------------------------------------------------------}
{                                                             }
{  Lesen einer Intel-Hex-Datei und Generieren der Original-   }
{  Eingabe-Datei. Achtung: 32 Bytes pro Zeile vorgeschrieben! }
{                                                             }
{  Programmaufruf:                                            }
{  READHEX <Eingabedatei> <Ausgabedatei>                      }
{                                                             }
{  <Ausgabedatei> = "LST:" ---> Drucker                       }
{                                                             }
{  Jürgen Plate 31.12.1986                                    }
{                                                             }
{-------------------------------------------------------------}

const
  MaxBuffer = 8191;         { Dateipuffer }
  CR = #13;
  LF = #10;

type
  HexByte   = String[2];
  HexZahl   = String[4];
  Zeile     = String[80];
  TBuffer   = array [0..MaxBuffer] of Byte;

var
 Infile      : Text;              { Eingabedatei (*.HEX)     }
 Outfile     : File;              { Ausgabedatei             }
 Prot        : Text;              { Fehlerprotokoll          }
 Buffer      : TBuffer;           { Eingabe-Puffer           }
 CPos,                            { Zeichenpos. i. d. Zeile  }
 I, J,                            { Zähler                   }
 Adr, NextAdr,                    { Adressen der Daten       }
 Checksum,                        { Prüfsumme                }
 Recs,                            { Recordzahl f. Blockwrite }
 Len,                             { Anzahl Bytes im Satz     }
 BufPos,                          { Pufferzeiger i. Ausgabep.}
 WarnCount   : Integer;           { Zähler für Warnungen     }
 Errcount    : Integer;           { Fehlerzähler             }
 Ch          : Char;              { Hilfsvariable            }
 H2          : Hexbyte;
 H4          : HexZahl;
 Eingabe,                         { Eingabezeile             }
 AktZeile,                        { geprüfte Eingabezeile    }
 Errors,                          { Fehlerzeilen             }
 ErrText     : Zeile;
 EndFile,                         { Eingabeende              }
 Error       : Boolean;           { Fehlerflag               }
 X, Y, Line  : Integer;           { Anzeigesteuerung         }

Function Hex_Byte(X : Integer): HexZahl;
  { Erzeugt aus einem Byte eine 2-stellige Hexzahl }
  const
    HexZiffer : Array [0..15] of Char = '0123456789ABCDEF';
  begin
  Hex_Byte := HexZiffer[Lo(x) shr 4]+HexZiffer[Lo(X) and $F];
  end;

Function Byte_Hex(H: HexByte): Byte;
  { Wandelt 2-stellige Hexzahl in ein Byte um }
  var
    B, C : Integer;
  begin
  Val('$'+H,B,C);
  Byte_Hex := B and $FF;
  end;

Function Word_Hex(X : HexZahl): Integer;
  { Wandelt 4-stellige Hexzahl in einen Integerwert um }
  var
    B, C : Integer;
  begin
  Val('$'+X,B,C);
  Word_Hex := B;
  end;

Procedure Nextline(var E: Zeile);
  { Lesen der nächsten Zeile aus der HEX-Datei,
    Zeilenlänge maximal 74 Zeichen --> maximal 32 Bytes (Hex) je Zeile }
  const
    Lastch : Char = ' '; { letztes gelesenens Zeichen }
  var
    ch : Char;
  begin
  E := '';
  if Lastch <> ':' then
    while (ch<>':') and not eof(Infile) do
      Read(Infile,ch); { Scan for ':', if line destroyed }
  repeat
    Lastch := ch;
    read(Infile,ch);
    if ch in [#32..#127] then
      E := E + Upcase(Chr(ord(ch) and $7F));
        { Bit 8 weg, in Großbuchstaben wandeln }
  until (length(E)=74) or (ch=CR) or (ch=LF) or eof(Infile);
  while length(E)<74 do E := E+'0';
  end;

Procedure Checkline(E : Zeile; Var A: Zeile);
 { Zeile auf falsche Zeichen prüfen, diese durch '0' ersetzen }
  var
    I : Integer;
  begin
  A := E;
  for I:= 1 to length(A) do
    if not (A[I] in ['0'..'9','A'..'F']) then
      begin
      A[I] := '0'; Errors[i] := '?';
      Error := True; Errcount := Errcount+1;
      Errtext := 'Falsche(s) Zeichen, ';
      end;
  end;

begin
LowVideo;
clrscr;
Writeln;
Writeln('═══════════════════════════════════════════════════════');
Writeln(' READHEX - Lesen von Dateien im Intel-Hexformat ');
Writeln('       Copyright (c) Franzis-Verlag 1987         ');
Writeln('═══════════════════════════════════════════════════════');
{ Kommandoparameter bearbeiten, Dateien eröffnen }
if ParamCount<>2 then
  begin
  Writeln('Parameterfehler: READHEX INFILE OUTFILE');
  halt;
  end;
```

Bild 2. Dieses Programm bearbeitet die empfangene Hex-Datei

```
{$I-}
Assign(Infile,Paramstr(1)); reset(Infile);
if IOResult<>0 then
  begin
  Writeln('Eingabedatei nicht vorhanden!');
   halt;
   end;
Assign(Outfile,Paramstr(2)); Rewrite(Outfile);
if IOResult<>0 then
  begin
  Writeln('Ausgabedatei kann nicht geöffnet werden!');
  halt;
  end;
Assign(Prot,'READHEX.ERR'); Rewrite(Prot);
{ Datei konvertieren }
Line := 0; EndFile := false; ErrCount := 0; WarnCount := 0;
NextAdr := $0100; BufPos := 0; Recs := (MaxBuffer+1) div 128;
Fillchar(Buffer,Sizeof(Buffer),^Z); { für Textdateien }
Writeln('Generiere "',Paramstr(2),'" aus "',Paramstr(1),'"');
Write('Zeile Nummer: '); X := WhereX; Y := WhereY; Writeln(Line:6);
Writeln('═══════════════════════════════════════════════');
while not (eof(Infile) or EndFile) do
  begin
  Line := Succ(Line); GotoXY(X,Y); Write(Line:6);
  Fillchar(Errors,Sizeof(Errors),' '); Errors[0] := chr(74);
  Errtext := '';
  Nextline(Eingabe);
  Checkline(Eingabe,AktZeile);
  CPos := 1; Error := false; Checksum := 0;
  H2 := Copy(AktZeile,CPos,2);
  Len := Byte_Hex(H2); { Längenangabe }
  Checksum := Checksum+Len;
  if H2='00' then { Dateiendesatz }
    EndFile := true
  else
    begin
    CPos := CPos+2;
    H4 := Copy(AktZeile,CPos,4);
    Adr := Word_Hex(H4);
    Checksum := Checksum+(Adr div 256);
    Checksum := Checksum+(Adr mod 256);
    if Adr<>NextAdr then { Adressen nicht fortlaufend }
      begin
      Errors[CPos] := '^';
      Error := True; WarnCount := WarnCount+1;
      Errtext := Errtext+'Warnung: '+H4+' <> '+
                 Hex_Byte(hi(NextAdr))+Hex_Byte(lo(NextAdr))+', ';
      NextAdr := Adr;
      end;
    CPos := CPos+4;
    H2 := Copy(AktZeile,CPos,2);
    if H2<>'00' then
      begin
      Errors[CPos] := '^';
      Error := True; Errcount := Errcount+1;
      Errtext := Errtext+'Falsche Satzart, ';
      end;
    Cpos := CPos+2;
    for j:=0 to Len-1 do { Bytes konvertieren }
      begin
      H2 := Copy(AktZeile,CPos,2);
      Buffer[BufPos] := Byte_Hex(H2);
      Checksum := Checksum+Buffer[BufPos];
      BufPos := BufPos+1; Cpos := CPos+2;
      if BufPos>MaxBuffer then { Buffer voll, wegschreiben }
        begin
        BlockWrite(Outfile,Buffer,Recs);
        Fillchar(Buffer,Sizeof(Buffer),^Z);
        BufPos := 0;
        end;
      end;
    Checksum := (256-Lo(Checksum)) and $00FF;
    H2 := Copy(AktZeile,CPos,2);
    If (Checksum)<>Byte_Hex(H2) then
      begin
      Errors[CPos] := '^';
      Error := True; Errcount := Errcount+1;
      Errtext := Errtext+'Prüfsummenfehler: '+H2+' <> '+Hex_Byte(Checksum);
      end;
    if Error then { Fehlerprotokoll ergänzen }
      begin
      Writeln(Prot,'Fehler in Zeile ',Line,': ');
      Writeln(Prot,':',Eingabe);
      Writeln(Prot,':',Errors);
      if Errtext <> '' then
        Writeln(Prot,'- ',Errtext);
      end;
    NextAdr := NextAdr+Len;
    end;
  end;
{ Letzen Buffer wegschreiben }
if BufPos>0 then
  BlockWrite(Outfile,Buffer,(BufPos div 128)+1);
Close(Infile);
Close(Outfile);
Close(Prot);

{ Abgesang }
Writeln; Writeln;
If (Errcount+WarnCount)=0 then
  begin
  Writeln('Keine Fehler beim Konvertieren aufgetreten.');
  Assign(Prot,'READHEX.ERR');
  Erase(Prot);
  end
else
  begin
  Writeln(Errcount,' Fehler gefunden.');
  Writeln(WarnCount,' Warnungen.');
  Writeln('Informationen in Datei READHEX.ERR.');
  end;
end.
```

```
Fehler in Zeile 21:
:200380004F4E3A22202D2G2D3E2042696C6473636869726D20202020202020202020202020DE
:                    ?                                                       ^
- Falsche(s) Zeichen, Prüfsummenfehler: DE <> EB
Fehler in Zeile 38:
:2005A0005829207D0D0A205075666665722020202020203A20496E4275666665723B202020D7
:                                                                            ^
- Prüfsummenfehler: D7 <> D8
Fehler in Zeile 128:
:2020E0006974656C6E3B2057726974656C6E3B0D0A656E642E0D0A1A00000000000000009C
:  ^
- Warnung: 20E0 <> 10E0,
```

Bild 3. Ein Beispiel für die Fehlermeldungen

genauso wie bei GENHEX. Die ersten Fehler werden bereits beim Einlesen eines Datensatzes (also einer Zeile) mit der Prozedur Nextline eliminiert. Zunächst wird der Satzanfang (":„) gesucht; verstümmelte Sätze führen also nicht zu Folgefehlern. Das achte Bit eines Bytes wird abgeschnitten und die Buchstaben werden in Großschreibung umgewandelt. In der Prozedur Checkline wird die eingelesene Zeile auf „verbotene" Zeichen untersucht, die dann durch eine „0" ersetzt werden. Gleichzeitig wird ein Fehlerprotokoll in der Datei READHEX.ERR angelegt. Danach wird der Hex-Satz in Bytes umgewandelt, wobei das Programm sich an der Längenangabe orientiert und Adreßangabe und Prüfsumme nachgeprüft werden. Alle Fehler führen zu einem Protokolleintrag in READHEX.ERR. In *Bild 3* ist ein Beispiel wiedergegeben. Der Fehler wird unter Angabe der Zeilennummer genau markiert und im Klartext erläutert. Die fehlerhaften Zeilen lassen sich später korrigieren. Die zurückgewonnenen Bytes werden in einem 8-KByte-Puffer eingetragen, der, wenn er voll ist, mit der Blockwrite-Prozedur auf die Ausgabedatei geschrieben wird.

Wenn sich das Programm mit der Nachricht „Keine Fehler aufgetreten" verabschiedet, kann man ziemlich sicher sein, die „Radiodaten" auch sicher auf Diskette zu haben.

Turbo-Symbolic-Debugger beschleunigt Fehlersuche

Turbo-Pascal bietet zur Suche logischer Programmfehler keinerlei Unterstützung. Üblicherweise läßt man sich in der Testphase Variableninhalte an den kritischen Stellen ausdrucken. Das bedeutet: Jedesmal, wenn man einen neuen Testpunkt setzt, muß man auch neu übersetzen. Bei längeren Programmen kann das sehr langwierig werden.

Lauer & Wallwitz, Wiesbaden, bietet jetzt mit dem Turbo-Symbolic-Debugger ein Programm für MS-DOS-Computer (ab Version 2.0) an, das die Testphase erheblich verkürzt. Es wird mit Hilfe eines Installationsprogramms so in die Turbo-Pascal-Software eingebunden, daß dort mit dem Befehl „Y" vom Hauptmenü aus eine neue Betriebsart – der Debugger – aufgerufen werden kann. Die wichtigste Eigenschaft des Debuggers: Nachdem ein Programm im Memory-Modus compiliert ist, lassen sich mit seiner Hilfe an beliebigen Stellen Breakpoints setzen. Erreicht das zu testende Programm nach dem Start einen solchen Breakpoint, wird es angehalten, und der Debugger ist aktiv. Jetzt kann man die Werte aller an dieser Stelle aktuellen Variablen ansehen und sogar ändern. Man kann neue Breakpoints setzen, das Programm fortsetzen oder die folgenden Befehle schrittweise ausführen.

Das sind zwar die wichtigsten Funktionen, aber das ist längst nicht alles. So kann der Entwickler bis zur Maschinenebene „herabsteigen", da man auch Variablenadressen, Speicher- und Registerinhalte ansehen kann. Komfortabel wird die Sache dadurch, daß man zu jeder Zeit auch Zugriff zum Quelltext hat. Will man das Programm mit neuen Breakpoints erneut starten, braucht man es nicht noch einmal zu kompilieren.

Wer, wie ich, von der üblichen Testmethode auf einen derart bequem zu handhabenden Debugger umsteigt, ist zunächst einmal begeistert. Vor allem, wenn es keinerlei größere Probleme gibt. Dennoch sind einige Situationen zu erwähnen, in denen die Freude nicht ungetrübt ist. Da der Debugger seine Daten normalerweise in einem Textfenster ausgibt, gibt es naturgemäß Schwierigkeiten, wenn man Grafikprogramme entwickelt. Statt des Fensters erscheint dann ein wirres Muster auf dem Bildschirm, und die Textausgabe funktioniert nicht richtig. Zwar läßt sich die Ausgabe auf den Drucker umleiten, aber nicht generell. Das wirre Muster bleibt also nach wie vor. Wer mit Overlays arbeitet, sollte den Debugger nur mit Vorsicht einsetzen. Das Handbuch empfiehlt, die Overlays auf jeden Fall zu entfernen, falls dies möglich ist.

Noch ein Wort zum Handbuch: Wer grundsätzlich weiß, was ein derartiges Programm leistet, wird keine Schwierigkeiten haben, schnell damit zurechtzukommen. Die Beschreibung ist ausführlich genug, und die Bedienung ist sehr logisch aufgebaut. Ein Neuling aber sucht vergeblich nach einer kleinen Einführung, die ihm sagt, wozu das Programm überhaupt gut ist. – Aber das stellt er nach kurzer Zeit ohnehin selbst fest.

Rudolf Hofer

Turbo-Pascal lädt DOS-3.3-Bilder

In mc 7/86 wurden zwei Turbo-Pascal-Prozeduren vorgestellt, mit deren Hilfe man die Möglichkeit hat, Apple-Grafiken abzuspeichern und zu laden. Will man mit Grafiken arbeiten, die unter DOS 3.3 erstellt worden sind, dann benutzt man am besten das Programm APDOS, um ein Bild als Binärdatei in das CP/M-format zu übertragen. Das Problem dabei ist, daß APDOS vor den eigentlichen Daten vier Bytes einfügt, die Startadresse und Dateilänge angeben. Diese vier Bytes müssen wieder entfernt werden, da sonst das Bild verschoben wird. Die im *Bild* dargestellte Prozedur ermöglicht es, ein mit APDOS übertragenes Bild auf den Schirm zu bringen und anschließend mit den in [1] beschriebenen Prozeduren (ohne die überflüssigen Bytes) zu speichern und zu laden.

Stefan Fischer

Literatur

[1 *Leyk, Heinz*: Turbo-Pascal: Hires-Bilder auf Diskette. mc 1986, Heft 7, Seite 71.

```
procedure pictransfer (name:dateiname);
 var f:file;
     feld:array[1..8192] of byte absolute $2ffc;

begin
  name:=concat(name,'.PIC');
  assign(f,name);
  reset(f);
  blockread(f,feld,64);
  close(f);
end;
```

Eine mit APDOS übertragene Grafik bringt diese Prozedur auf den Bildschirm des Apple-II

Herwig Feichtinger

AT-Kompatibler aus Taiwan

Multitechs Accel-900

Nach der Verfügbarkeit von Intels CPU 80286 in ausreichenden Stückzahlen überschwemmt uns jetzt eine Welle von preiswerten AT-Kompatiblen aus Fernost. Multitech, ein renommierter taiwanesischer Hersteller, begann vor Jahren seine Laufbahn mit Z80-Lernsystemen und bietet neben seiner PC-Serie nun ebenfalls einen AT namens Accel 900 unter 5000 DM an.

Unser Testgerät war mit einem TTL-Schwarzweiß-Monitor, 512 KByte RAM, einer 20-MByte-Festplatte und einem Floppy-Laufwerk ausgestattet. Als Video-Interface bauten wir eine MGA-Karte (Monochrome Graphics Adapter, Hercules-kompatibel) von Addonics ein (Typ VCA-311), die auch eine Parallelschnittstelle für Drucker enthält. Mitgeliefert wurde auch das Betriebssystem MS-DOS 3.1 von Microsoft. Eine zusätzliche SPB-AT-Karte von Multitech stellt eine zweite parallele sowie zwei serielle Schnittstellen zur Verfügung, von denen die erste (COM1) wohl aus Platzgründen leider nur als nicht CCITT-normgerechte neunpolige Buchse herausgeführt ist. Obwohl die V.24-Schnittstellen mit 16450- statt mit den vom IBM-PC bekannten 8250-Bausteinen arbeiten, sind sie mit denen anderer PCs und ATs völlig kompatibel – vorausgesetzt, man stellt die DIP-Schalter auf der Karte richtig ein, was im (englischen) Handbuch für Fachleute verständlich beschrieben ist.

Der Multitech-Monitor ist recht gut entspiegelt. Schrift und Grafik nutzen eine Bildschirm-Diagonale von 26 cm. Auf der Unterseite findet man eine rastbare Taste, mit der sich die Darstellung von negativ (weiß auf schwarzem Untergrund) auf positiv umschalten läßt. Mir gefällt die positive Darstellung recht gut, sie ist sehr augenfreundlich und man ist sie vom Papier her ohnehin gewohnt. Ein kleiner Haken: Die z. B. für Schriftmarkierungen verwendeten Grautöne gehen bei der Positiv-Darstellung leider verloren, lediglich das bei der MGA-Karte im Gegensatz zur Farbgrafikkarte mögliche Attribut „Unterstreichen" bleibt sichtbar.

Was bei manch anderem Hersteller noch ein PC kostet, dafür bekommt man bei Multitech schon einen AT-Kompatiblen

Komfort-Tastatur

Die Tastatur ist zwar offensichtlich preiswert gemacht (die CTRL-Taste klemmte, wenn man sie nicht in der Mitte betätigte), aber entspricht funktionell dem neuen IBM-Layout mit einem Cursor- und einem Ziffernblock. Der Zustand von Caps Lock, Num Lock, Scroll Lock und Betriebsspannung (Power) wird von Leuchtdioden angezeigt. Ausgesprochen verwirrend ist die Tastenbeschriftung. Die US-Belegung ist in schwarzer Schrift eingraviert, und jene Tasten, die im deutschen Modus andere Zeichen liefern, enthalten zusätzlich eine blaue Gravur mit Umlauten und Sonderzeichen. Dieser Testbericht wurde auf dem Accel-900 geschrieben, und viele Male wurde dabei die Backspace-Taste gedrückt ...

Die Doppelkreuz-Taste ist merkwürdigerweise zweimal vorhanden, einmal neben der ALT-Taste und einmal neben Backspace. Beide Tasten liefern den gleichen Code. Im Textsystem Euroscript wurde dabei statt des Doppelkreuzes allerdings ein Kleiner-Zeichen angezeigt, und die Kleiner-Taste (links oben) lieferte dafür das Doppelkreuz. Auf der Betriebssystem-Ebene entsprach die Belegung dagegen dem Tastenaufdruck. (Zum Glück läßt Euroscript ein Umdefinieren der Tastenbelegung zu.)

Da sowohl ein Cursor- als auch ein Ziffernblock vorhanden ist, wäre es naheliegend gewesen, die Zifferntasten nach dem Einschalten gleich in den Num-Lock-Modus zu versetzen, wie es manche Hersteller tatsächlich tun. Aber das ist vielleicht eine Frage der Gewöhnung.

Zehnmal schneller als ein IBM-PC

Die Zentraleinheit mit der 80286-CPU (10 MHz Taktfrequenz) liefert nach Peter Nortons Programm „Sysinfo" einen Faktor von 10,7 über der Geschwindigkeit

eines herkömmlichen 8088-PC mit 4,77 MHz Taktfrequenz. Gerade beim Compilieren oder Assemblieren umfangreicherer Programme macht sich der Zeitvorteil drastisch bemerkbar. Der mc-Mini-Benchmark (2000 Divisionen 10/3 in GW-Basic) läuft in 1,5 Sekunden ab (IBM-PC: etwa sieben Sekunden). Hier ist der Vorsprung des AT also nicht ganz so groß, wie Nortons Sysinfo glauben machen könnte. Überhaupt darf man die Ergebnisse des Norton-Programms, die in Testberichten und in der Anzeigenwerbung gern als Benchmark-Ersatz verwendet werden, nicht verallgemeinern; Norton läßt den Prozessor lediglich in einer Schleife mit ein paar Arithmetik-Befehlen laufen und mißt die Zeit unter Zuhilfenahme des Timers. Andere Anwenderprogramme verwenden möglicherweise eine andere Zusammensetzung von Befehlen und führen deshalb auch nicht unbedingt zu den gleichen Ergebnissen.
Manche Programme lassen sich aber ohnehin nur auf einem AT-Kompatiblen sinnvoll einsetzen – entweder, weil sie in der falschen Programmiersprache geschrieben sind (wie etwa das auf PCs viel zu langsame Textprogramm „Easy" von Micropro, das in Modula programmiert ist) oder weil sie mit jeder Menge Overlays arbeiten (z. B. „Windows" von Microsoft).
Eingebaut sind auch eine batteriegepufferte Echtzeituhr, die sich lediglich alle vier Jahre um einen Tag irrt, weil sie keine Schaltjahre kennt, sowie ein kleines CMOS-RAM, in das vom mitgelieferten Installationsprogramm die Systemparameter geschrieben werden: installierte Laufwerke, Art des Monitors, Speichergröße, CPU-Taktfrequenz und so weiter.
Die Zentraleinheit ist optisch und akustisch „beeindruckend" – zum einen wegen ihrer schreibtisch-raubenden Größe (16 cm hoch, 41,5 cm tief, 52,5 cm breit), zum anderen wegen des unüberhörbaren Geräusches von Lüfter und Festplatte. Es gibt inzwischen sowohl leisere als auch kleinere AT-kompatible Rechner. Die mechanische Ausführung ist hingegen durchaus zu loben: Das Metallgehäuse sorgt nicht nur für Stabilität, sondern auch für eine nur geringe Störstrahlung des Rechners. Lediglich die Frontseite besteht aus Kunststoff-Formteilen. Wie bei allen AT-Kompatiblen (und manchen Möchtegern-PCs) ist ein Schlüsselschalter zum Absperren der Tastatur vorhanden. Neben ihm sitzt eine gefährlich leicht erreichbare Reset-Folientaste: Ein falscher Griff, und alles bisher eingetippte ist verloren ...

Einer der insgesamt acht Slots ist bereits ab Werk mit der CPU-Karte belegt, ein zweiter nimmt die Video-Karte auf. Die Schnittstellen-Karte SPB-AT belegt an der Rückwand wegen der nötigen Buchsen zwei Slots, innen jedoch nur einen.

Mit 1,2-MByte-Laufwerk

Unser Testgerät war mit einem 1,2-MByte-Diskettenlaufwerk für High-Density-Disketten sowie einer 20-MByte-Festplatte ausgestattet. Verwöhnt durch einstmals gute Erfahrungen mit dem Osborne-6 wurde entgegen den Empfehlungen des Multitech-Handbuchs versucht, damit auf Disketten zu schreiben, die auf einem anderen PC für 360 KByte Kapazität formatiert wurde; bei kurzen Dateien ging das ein paarmal auch gut, einmal sagte der PC beim Zurücklesen allerdings „Data Error reading Drive A".

Denn während es beim Lesen von 360-KByte-Disketten auf dem 1,2-MByte-Laufwerk keine Schwierigkeiten gibt, ist die Kopfbreite von letzterem geringer, so daß nicht die volle Spurbreite beschrieben werden kann. Da dieser Testbericht auf dem Accel-900 geschrieben wurde, anschließend aber im 360-KByte-Format an die Setzerei gehen sollte, blieb nichts weiter übrig, als Shamrock's Low-Cost-Netzwerk NET.24 zu bemühen, um Fehler auszuschließen. (Ähnliche Überspiel-Probleme können übrigens auch bei Rechnern mit 720-KByte-Laufwerken auftreten, z. B. beim Siemens-PCD oder bei einem entsprechend ausgerüsteten M-24 von Olivetti.)

Übrigens erfordert das 1,2-MByte-Format auch entsprechende High-Density-Disketten. Der Versuch, es einmal mit normalen doppelseitigen Disketten doppelter Dichte zu probieren, führte schon beim Formatieren zu zahlreichen fehlerhaften Sektoren. Man kann nun sagen, dann ist ja immer noch mehr Platz drauf als im 360-KByte-Format – nur, das Kopieren einer solchen Diskette mit DISKCOPY ist dann leider unmöglich, weil DISKCOPY bei einem fehlerhaften Sektor sofort abbricht. Außerdem sei dahingestellt, ob man die Daten auch beim nächsten Mal noch sicher lesen kann.

Software und Dokumentation

Wie schon erwähnt, wurde mit unserem Testgerät das Betriebssystem MS-DOS 3.1 mitgeliefert. Multitech ergänzte es um einige eigene, systemspezifische Programme, die u. a. der Systemkonfiguration dienen und die menüorientiert,

Technische Daten

Hersteller: Multitech (Taiwan)
Name: Accel-900
Prozessor, Takt: 80286 (Intel), 10 MHz
Mitgelieferte Software: MS-DOS 3.1
Floppy-Laufwerke: 1 oder 2
Festplatten: bis zu 2
Slots: 7
Video-Adapter: CGA-, MGA- (Hercules) oder EGA-Karte
Echtzeituhr: eingebaut
Handbücher: Benutzerhandbuch deutsch, MS-DOS-Manual englisch

allerdings englischsprachig, arbeiten. Ein Basic-Interpreter, der bei anderen Herstellern zum Standard-Lieferumfang gehört, ist nicht dabei. Das GW-Basic anderer Anbieter läuft jedoch ohne Probleme auch auf dem Accel-900, wenn man von den üblichen Problemen mit der Grafik-Darstellung auf Hercules-kompatiblen Video-Interfaces absieht (GW-Basic kann normalerweise Grafik nur auf dem Color-Graphics-Adapter darstellen).
Irgendwelche Kompatibilitäts-Schwierigkeiten hinsichtlich ROM-BIOS oder Hardware-Adressen konnten nicht festgestellt werden; getestet wurde dies u. a. mit hardware-nahen Programmen wie Turbo-Pascal (Borland), GW-Basic, Quickbasic und Windows (Microsoft), Euroscript (NAS), EDI, PCTERM und NET.24 (Shamrock). Der Microsoft-Flugsimulator hätte eine Farbgrafik-Karte erfordert, mit der MGA-Karte ist er nicht lauffähig.
Die mitgelieferte Dokumentation bestand aus einem deutschsprachigen Benutzer-Handbuch, das im wesentlichen die Installation und den elementaren Umgang mit MS-DOS beschreibt. Es enthält auch die Belegung aller Stecker sowie die vorwiegend für Programmierer wichtigen Tastatur-Scan-Codes. Das zu MS-DOS 3.1 gehörende ausführlichere Handbuch steht hingegen nur englischsprachig zur Verfügung. Für den kommerziellen Normalbenutzer ist das aber sicher kein zentrales Problem, wenn auch der Gesamtumfang und die Qualität der Handbücher mit den von IBM, Olivetti, Zenith und anderen Herstellern gesetzten Standards nicht ganz mithalten können.
Fazit: Alles in allem ein preiswerter AT-Kompatibler, der durch solide Verarbeitung und hohe Geschwindigkeit glänzt.

Dieter Strauß

Periscope

Ein symbolischer Debugger für den PC

Fast jeder Programmierer, der Assemblerprogramme für den IBM-PC oder kompatible Computer geschrieben hat, dürfte bereits mit dem DEBUG-Programm gearbeitet haben. Wer hat sich während der Fehlersuche nicht schon gewünscht mit Symbolen, wie sie in der ASM-Quelldatei verwendet werden, zu arbeiten, statt sich mit Adressen in Hex-Codierung herumzuplagen? Mit Periscope, einem symbolischen Debugger, geht dieser Wunsch in Erfüllung. mc hat Periscope auf dem Commodore PC 10 getestet.

Das Periscope-Debugger-System besteht in der Periscope-I-Version aus einer PC-Erweiterungskarte mit 16 KByte RAM, einem Unterbrechungsschalter, einer Programmdiskette, einem Faltblatt mit der Befehlsübersicht und einem Handbuch. Es wird von H&B EDV, Tettnang, vertrieben.

Das eigentliche Debugger-Programm wird resident in den Speicher geladen und kann mit dem Unterbrechungsschalter aufgerufen werden. Periscope versteht die Maschinensprache der Typen 8088, 8086 und 80286 im 8086-Modus. Da die derzeitig erhältlichen DOS-Versionen den virtuellen Adreß-Modus der 80286 nicht verwenden, wäre es nicht sinnvoll, diese Betriebsart mit einem Debugger zu unterstützen.

Die PC-Erweiterungskarte wird vom Hersteller Submarine-Karte genannt. Ihr Speicher ist schreibgeschützt, vom Speicherrefresh des PCs unabhängig und kann nur durch die E/A-Ports angesprochen werden. Die Startadresse des Speichers kann mit DIP-Schaltern in 64-KByte-Schritten eingestellt werden.

Die Portadresse ist in 4-Byte-Schritten einstellbar. Werksseitig ist die Submarine-Karte auf die Segment-Adresse D000H und die Port-Adressen 300H und 301H eingestellt. Um die Karte für einen Schreibzugriff vorzubereiten, muß die Periscope-Software DBH in das erste Port schreiben. Während unserer Testreihe wurde der Speicher der Submarine-Karte bei keinem Systemabsturz überschrieben.

Ein Rettungsanker bei Systemabstürzen

Der Unterbrechungsschalter ist in einem kleinen handlichen Gehäuse untergebracht und wird zusammen mit einem ungefähr 1 m langen Kabel geliefert. Nachdem die Submarine-Karte im PC eingebaut ist, wird der Unterbrechungsschalter mit der Cinchbuchse der Submarine-Karte verbunden. Der Unterbrechungsschalter erzeugt, sobald er gedrückt wird, einen nicht maskierbaren Interrupt (NMI). Vorausgesetzt, Periscope ist geladen, kann damit die Kontrolle an den Debugger auch bei Systemabstürzen zurückgegeben werden.
Für alle Leser, die sich jetzt bereits überlegen, auf welche Assemblerprogramme man sich zur Fehlersuche stürzen kann, folgt jetzt noch ein kleiner Wermutstropfen – die gesamte Dokumentation ist leider nur in Englisch erhältlich.
Da die Befehle des Periscope-Systems den DEBUG-Befehlen ähnlich sind, haben wir es innerhalb kürzester Zeit bedienen können. Unter Periscope können Fenster definiert werden, um den disassemblierten Objektcode, den Stapel, die Register und die Daten anzuzeigen. Speicherinhalte können in verschiedenen Formaten wie ASCII-, Byte-, Integer-, Wort- und Doppelwort-Format angezeigt werden. Soll der Quellcode auf dem Bildschirm dargestellt werden, muß nur das Kommando V (View File) eingegeben werden.
Mit der Funktionstaste F10 kann zwischen der Debugger-Anzeige und der Programmausgabe hin- und hergeschaltet werden.

Auf der Periscope-Programmdiskette befindet sich ein Beispielpogramm, um mit Periscope die ersten Gehversuche zu machen. Periscope verwendet die vom DOS-Linker erzeugte MAP-Datei, um Adressen mit leicht zu merkenden Symbolen zu ersetzen. Wird in einem Programm ein Unterprogramm namens check_parameter verwendet und der erste Aufruf dieses Unterprogramms gesucht, so muß nur G .check_parameter (G steht für Go) eingegeben werden. Da in die Speicherstelle des Breakpoints CCH eingefügt wird, können in ROMs keine Breakpoints gesetzt werden. Das ursprüngliche Byte wird wiederhergestellt, wenn die Kontrolle an Periscope zurückgegeben wird.

Zahlreiche Optionen zum Installieren des Debuggers

Um Periscope zu installieren wird PS, gefolgt mit den gewünschten Installierungsoptionen, eingegeben.

Zum Installieren von Periscope wird eine große Anzahl an Optionen angeboten:

/A – Sowohl ein monochromer als auch ein Farbbildschirm sind am System über verschiedene Adapterkarten angeschlossen. Der Debugger benutzt den Bildschirm, der nicht gerade aktiv ist.

/B:nn – Setzt die Größe des Traceback-Puffers auf Werte zwischen 0...16 KByte. Standard ist 1 KByte. nn gibt die Zahl der KByte in hexadezimaler Notation an.

/C:nn – Das Farbattribut für den Debugger-Bildschirm ist einstellbar.

/D – Vor dem Booten wird der ursprüngliche INT 13H (Disketten-E/A) wiederhergestellt.

/E:nn – Die Größe des Quelldatei-Puffers läßt sich in dem Bereich 0...16 KByte einstellen.

/F – Die Ausgabe auf den Farbbildschirm wird beschleunigt. Diese Option ist nur für die Grafikkarten sinnvoll, bei denen in den Bildschirmpuffer geschrieben werden kann, ohne auf dem Bildschirm Störungen hervorzurufen.

/H – Die Datei PSHELP.TXT wird resident in den RAM geladen. Ist diese Datei geladen, so können jederzeit Hilfstexte angezeigt werden.

/I:nn – Wird diese Option installiert, so kann Periscope auf andere speicherresi-

dente Programme zugreifen. Es sind die Interrupts 60H...FFH zugelassen.

/J – Auf dem AT kann die Sys-Req-Taste zum Aufruf von Periscope verwendet werden.

/K – Shift-PrtSc soll zum Aufruf von Periscope verwendet werden.

/L:nnnn – Periscope wird normalerweise im Speicher so niedrig wie möglich geladen. Wenn Fehler in Gerätetreiber gesucht werden sollen, kann der Debugger in einen höheren Speicherbereich geladen werden. nnnn gibt das Segment an, ab der Periscope geladen werden soll.

/M:nnnn – Wird Periscope in der Version mit der Submarine-Karte verwendet, so beginnt das schreibgeschützte Speichersegment mit Segmentadresse D000H. Mit dieser Option kann der Bereich geändert werden.

/P:nnn – Setzt die E/A-Ports der Submarine-Karte auf andere Adressen als die voreingestellten (300H und 301H). Die DIP-Schalter auf der Karte müssen entsprechend umgestellt werden.

/R:nn – Die Tabelle der Record-Definitionen wird auf einen anderen Wert als den voreingestellten 1 KByte gesetzt.

/S:nn – Der Bildschirmpuffer ist 4 KByte groß. Bei CGA-kompatiblen Karten wird nn = 10 gesetzt (10H = 16 KByte). Der Bildschirmpuffer kann bis 32 KByte groß werden.

/T:nn – Setzt die Symbol-Tabelle auf einen anderen Wert als 1 KByte. Diese Option wird für Programme mit einer großen MAP-Datei benötigt.

/V:nn – Periscope setzt die BIOS-Interrupt-Vektoren auf die Werte, mit denen sie nach einem Systemstart initialisiert werden. Soll Periscope modifizierte Vektoren verwenden, so muß mit nn die Nummer des Vektors eingegeben werden.

/W – Die Fenstergröße der angezeigten Daten, Register, Stapel und disassemblierten Befehle läßt sich mit dieser Option verändern.

/Z – Periscope soll nur den schreibgeschützten Speicher der Submarine-Karte belegen. Damit wird die Leistungsfähigkeit stark eingeschränkt, jedoch wird bei keinem Systemabsturz dieser Speicher überschrieben.

Zahlreiche Dienstprogramme erleichtern die Fehlersuche

Mit Periscope werden einige nützliche Dienstprogramme mitgeliefert.
PSPATCH.COM wird benötigt, wenn der verwendete PC nicht ganz kompatibel ist und die Einsprungpunkte seiner BIOS-Interruptvektoren vom IBM-Standard abweichen.

PUBLIC.COM erzeugt Public-Anweisungen im Assemblerquellprogramm. Mit RS.COM können selbst erstellte Dateien mit Record-Definitionen verifiziert werden. SYMLOAD.COM lädt die Symboltabelle von Periscope mit den Symbolen, die im Programm verwendet wurden.

TS.COM verifiziert die MAP-Datei und erzeugt auf Wunsch eine Periscope-Symbol-Datei. Periscope unterstützt außer den MAP-Dateien, die von Microsofts LINK-Programm erzeugt werden, LINK86 von Digital Research (DRI) und PLINK von Phoenix. TS.COM erzeugt aus Dateien mit Adreßreferenzen, die von den beiden letztgenannten Linkern gebildet wurden, eine von Periscope lesbare Datei.

Nachdem Periscope geladen ist, kann mit dem Debuggen begonnen werden. Das fehlerhafte Programm, welches als COM- oder EXE-Datei vorliegen muß, wird mit dem Befehl RUN, gefolgt vom Programmnamen und den Parametern der Kommandozeile, zur Ausführung gebracht. Die Kommandozeile ist die gleiche, als ob das Programm direkt von DOS aus gestartet würde. Für jedes Programm kann eine Datei (Erweiterung .DEF) angelegt werden, in der Tastenfunktionen umgeändert und Speicherbereiche als logisches Feld, wie z. B. ein PSP oder FCB, definiert werden. Ist im Directory keine zum Programm gehörende DEF-Datei vorhanden, so wird die mitgelieferte PS.DEF-Datei geladen. *Bild 1* zeigt den Aufbau der mitgelieferten Datei PS.DEF. Ab der Zeile mit \FCB wird der Aufbau des Dateisteuerblocks definiert. Wird in einem Programm unter Periscope DR CS:0 .PSP eingegeben, so erscheint auf dem Bildschirm der in *Bild 2* gezeigte Programm-Segment-Präfix.

Periscope versteht über 70 Kommandos. Ohne die jederzeit aufrufbare Befehlsübersicht wäre Periscope nur schwer bedienbar. Programmierer, die bereits längere Zeit mit Periscope gearbeitet haben und alle Befehle kennen, können beim Installieren die Datei mit der Befehlsübersicht weglassen, um die Größe des speicherresidenten Programms zu verringern.
Die Periscope-Kommandos können in mehrere Gruppen unterteilt werden: Befehle zum Anzeigen von Speicherinhalten und Registern (Dump)

```
\f1=k;dr cs:0 .psp;
\f2=dr cs:5c .fcb;
\f3=k;dr 0:0 .int0;
\f4=k;dr 0:40 .int10;
\f5=k;dr 0:80 .int20;
\f6=k;dr 0:C0 .int30;
\FCB                ; File Control Block
Drive,b,1           ; Drive 0=default, 1=A, 2=B, etc.
File,b,8            ; File name
Ext,b,3             ; File extension
Block #,w,2         ; Current block number
Rec Size,w,2        ; Logical record size
File Size,d,4       ; File size
Date,w,2            ; Date of last update
Res.,b,a            ; Reserved for DOS
Rec #,b,1           ; Current relative record number
Rel Rec #,d,4       ; Relative record number from beginning of file
\PSP                ; Program Segment Prefix
Int 20,b,2          ; DOS return
Topmem,w,2          ; Amount of memory in paragraphs
Res.,b,1            ; Reserved for DOS
Long Call,b,1       ; Long call to DOS function dispatcher
DOS Func,d,4        ; CS:IP of DOS function dispatcher
Term Addr,d,4       ; CS:IP of DOS terminate address
Brk Addr,d,4        ; CS:IP of Ctrl-Break exit address
Err Addr,d,4        ; CS:IP of critical error exit address
Res.,b,16           ; Reserved for DOS
Environ,w,2         ; DOS 2.00 Environment segment
Res.,b,2e           ; Reserved for DOS
PSP1,b,10           ; The first PSP read from the command line
PSP2,b,14           ; The second PSP read from the command line
```

Bild 1. In der PS.DEF-Datei sind Funktionstasten-Makros, der Dateisteuerblock und der Programm-Segment-Präfix als Record definiert

```
AX=0000  BX=0000  CX=01A0  DX=0000  SP=FFFE  BP=0000  SI=0000  DI=0000
DS=1F31  ES=1F31  SS=1F31  CS=1F31  IP=0100   NV UP EI PL ZR NA PE NC
                START:
1F31:0100 E98D00        JMP     EINGABE
>dr cs:0 .psp
Int 20      CD 20
Topmem      4000
Res.        00
Long Call   9A
DOS Func    F00D:FFF0
Term Addr   0747:028F
Brk Addr    0747:029C
Err Addr    0747:05E8
Res.        47 07 01 01 01 00 02 FF FF FF FF FF FF FF FF FF
            FF FF FF FF FF FF
Environ     1F2E
Res.        38 09 CB 20 00 00 00 00 00 00 00 00 00 00 00 00
            * 0001 Lines Of 00 Skipped *
            00 00 CD 21 CB 00 00 00 00 00 00 00 00 00
PSP1        00 31 20 20 20 20 20 20 20 20 20 00 00 00 00
PSP2        00 37 30 20 20 20 20 20 20 20 20 00 00 00 00
            00 00 00 00
>
```

Bild 2. Periscope zeigt den Programm-Segment-Präfix eines Programms an

Breakpoint-Optionen
Trace-Befehle
Suchbefehle
Assemblier- und Disassemblierbefehle
Editierbefehle
Registermanipulationsbefehle

Das Kommando „Breakpoint on Byte" ist ein gutes Beispiel, um zu zeigen wie einfach Periscope bedient werden kann. Mit „BB .LINE_COUNT EQ 1F" wird ein Byte-Breakpoint gesetzt, wenn die Variable LINE_COUNT den Wert 1FH annimmt.

Unsere Testreihe hat ergeben, daß Periscope in der Händen eines Programmierers ein gutes Werkzeug ist, um in Programmen schneller Fehler zu finden.

Periscope ist in drei Versionen erhältlich. Periscope II wird ohne Submarine-Karte jedoch mit dem Unterbrechungsschalter geliefert. Eine reine Softwarelösung ist Periscope II-X. Periscope II und II-X werden bei Programmabstürzen überschrieben. Es ist dennoch komfortabler mit Periscope II und II-X zu arbeiten als mit dem DEBUG-Programm.

Tips zu MASM und LINK

Es ist beim Erstellen eines größeren Assembler-Programms eine gute Angewohnheit, dieses in mehrere kleine Module aufzuteilen. Der Quellcode jedes Programm-Moduls kann einzeln geschrieben und editiert werden. Das Quellprogramm (Source Code) muß als ASCII-Textdatei auf einer Diskette oder der Festplatte abgespeichert werden.

Überlicherweise werden Quellcodedateien im PC-Bereich mit der Erweiterung .ASM gekennzeichnet.

Im nächsten Schritt wird das Quellprogramm vom Übersetzerprogramm, dem Assembler, in Maschinensprachebefehle umgesetzt. Das Ergebnis dieser Übersetzung wird Objektcode genannt. Objektcodedateien werden mit der Dateinamenerweiterung .OBJ gekennzeichnet. Programme im Objektcode-Format können jedoch noch nicht ausgeführt werden. Das Objektcode-Format ist so aufgebaut, daß mehrere Objektcode-Module zu einem einzigen größeren Programm zusammengefaßt werden können. Der Microsoft-Macro-Assembler (MASM) erzeugt beim Übersetzungslauf außer der Objektcode-Datei eine .LST-Datei. In der .LST-Datei stehen der Quellcode mit den Kommentaren, die Zeilennummern, die relativen Adressen, der Maschinencode im Hex-Format und, solange das Programm noch nicht von Syntax-Fehleren befreit ist, die Fehlermeldungen. Am Ende der .LST-Datei sind alle verwendeten Symbole wie Segmentbezeichnungen, Konstanten und Variablen aufgeführt.

Konstanten werden mit dem Ausdruck Number, gefolgt von dem Wert der Konstanten, gekennzeichnet. Mehrere Module im Objektcode-Format werden mit einem Binderprogramm zu einem ausführbaren Programm gebunden. In der DOS-Welt heißt der Binder LINK. Der Binder, häufig Linker genannt, führt außer dem Bindeprozeß noch die Umwandlung des Objektcode-Formats in das EXE-Format durch. Das DOS-LINK-Programm wird mit LINK aufgerufen. In der Kommandozeile von Link dürfen Parameter stehen. Der erste Parameter enthält die Liste der Programm-Module im Objektcode-Format. Als Begrenzer-

```
Warning: No STACK segment

 Start  Stop   Length  Name
 00000H 002A2H 02A3H   CODE

 Origin   Group

  Address          Publics by Name

 0000:018F        ANZEIGE
 0000:01CA        CHECK_PARAMETER
 0000:017B        DISPLAY
 0000:0177        ENDE
 0000:0252        FEHLER_HINWEIS
 0000:028D        FEHLER_TEXT
 0000:0008  Abs   GRAFIK
 0000:01A9        INSTALLATION
 0000:0053  Abs   INT_VEKTOR
 0000:02A1        PARA1
 0000:02A2        PARA2

Program entry point at 0000:0100
```

Die .MAP-Datei enthält wichtige Informationen für den Programmierer

zeichen wird in der Liste der Objekt-Module das Pluszeichen verwendet. In der Liste der Objekt-Module dürfen auch von einem Compiler erzeugte Module stehen, sofern der Compiler sich an die Konventionen des DOS-Linkers hält. Die derzeit erhältlichen Versionen von Turbo-Pascal erzeugen jedoch keinen linkfähigen Code. In Turbo-Pascal muß man sich mit der Inline-Anweisung behelfen.

Der zweite Parameter der Link-Kommandozeile gibt den Namen des gebundenen Programms an und der dritte bezeichnet die Ausgabeeinheit für Meldungen. Der vierte Parameter gibt den Namen der Bibliotheks-Dateien an, die durchsucht werden sollen. Er kann von mehreren Schaltern gefolgt werden. Wird als Schalter /M eingegeben, so erzeugt der Linker eine .MAP-Datei, in der die gesamten Adreßreferenzen des Programms stehen. Die Einträge der .MAP-Datei enthalten die Segmentnamen der Module, die Segmentlänge, den Wert der verwendeten Symbole und ihre Adressen im ablauffähigen Programm. Unser Bild zeigt die .MAP-Datei eines Programms, das nur aus einem Objekt-Modul besteht. *Dieter Strauß*

Frank Koppensteiner

MS-DOS als Beifahrer

Amiga-Sidecar von Commodore

Eigentlich heißt Sidecar nichts anderes als Beiwagen. Beim Amiga steht diese Bezeichnung für eine Erweiterung, die an der Bus-Schnittstelle angeschlossen wird. Der Amiga 1060 Sidecar, wie die offizielle Bezeichnung lautet, ist ein vollständiger MS-DOS-Rechner, wodurch dem Amiga mit einem Schlag die ganze Palette der MS-DOS-Programme offensteht.

Vielleicht erinnern sich einige an Commodores Versuch, mit einem 5¼-Zoll-Laufwerk und einem Programm, das den Prozessor 8086 emulierte, den Amiga zu einem PC-kompatiblen Single-Task-Computer zu degradieren. Selbst der Prozessor 68000 im Amiga ist jedoch mit einer solchen Aufgabe überfordert, und so war das Zeitverhalten dieser Lösung kaum überzeugend.

Hardware statt Software

Die Hauptplatine des Sidecar ist nach dem gleichen Muster aufgebaut wie die vieler PC-Kompatibler (nur daß sie hier weit weniger Platz braucht als etwa beim Commodore PC-10): Ein Prozessor vom Typ 8088 mit 4,77 MHz Taktfrequenz, eigenes RAM, eigene Steckplätze und ein eigenes Laufwerk. Das Einzige, was diesem MS-DOS-Rechner fehlt, sind der Bildschirm und die Tastatur, deren Aufgaben vollständig vom Amiga übernommen werden. Die Steuerung und Kontrolle des Sidecar erfolgt über die Amiga-Tastatur, die Darstellung der Ausgabe unter Ausnutzung aller Amiga-Fähigkeiten auf dessen Bildschirm.

Der PC stellt hier eine eigene Task dar und läuft, wie jede andere Amiga-Task, in einem eigenen Fenster, daß mit dem Maus-Cursor aktiviert werden kann.

Man kann das PC-Fenster genauso verkleinern wie jedes andere, also den gerade eine Datenbank sortierenden PC im Briefmarkenformat „in die Ecke" stellen oder hinter anderen Fenstern verschwinden lassen. Bei Bedarf holt man ihn dann wieder hervor und widmet sich seinen Resultaten.

Die Installation ist denkbar einfach: Vorausgesetzt werden mindestens 512 KByte RAM im Amiga, der PC wird dann einfach rechts am Amiga eingesteckt und schließt bündig mit der Vorderfront ab, hinten bleibt eine Überlänge von 5,5 cm. Der Amiga-Bus ist nicht durchgeführt (man kann also nach dem Sidecar keine anderen Erweiterungen mehr anschließen), jedoch die beiden Stecker für Maus und Joystick. Das Netzkabel des Amiga wird ins Sidecar gesteckt, von dem wiederum ein Kabel zum Amiga führt, so daß beide Systeme mit einem einzigen Netzschalter einzuschalten sind. Das soll auch so sein, denn laut Anleitung darf der Amiga nicht alleine abgeschaltet werden.

Um keine falsche Hoffnungen zu wekken: Sidecar muß direkt neben dem Amiga stehen, zumindest kann man den verfügbaren Hardware-Informationen des Herstellers entnehmen, daß der Amiga-Bus nicht wie ein C-64-1541-Kabel problemlos mißhandelt werden kann.

Inbetriebnahme

Im Lieferumfang des Sidecar ist das Programm Kickstart 1.2 enthalten, außerdem die Workbench-Version 1.2, mit denen das ganze System gestartet wird. In das PC-Laufwerk wird eine ganz normale MS-DOS-Diskette eingelegt. Der Kickstart-Ladevorgang verläuft wie gewohnt, beim Laden der Workbench werden dann die neuen Treiber-Programme eingebunden, ähnlich der Einbindung des Config.Sys im MS-DOS-Bereich. Die Amiga-Treiber für Sidecar heißen übrigens sinnigerweise „Janus". Nach dem Laden erkennt der Amiga die Erweiterung und sendet zunächst einen Reset-Impuls, woraufhin Sidecar einen Speichertest ausführt und das Betriebssystem MS-DOS von seiner Diskette lädt.

Wer das Laden der Workbench mit CTRL-D vermeidet, landet wie gewohnt im Kommando-Interpreter (CLI), ansonsten wird die Workbench angezeigt.

Mittlerweile liegen übrigens Kickstart 1.2, Workbench 1.2 und Amiga-Basic 1.2 für alle Amiga-Besitzer bei den Commodore-Händlern vor.

In der Workbench oder, im Kommando-Interpreter, mit List erscheinen drei neue Programme, und zwar „PC mono", „PC color" und „lpt1". Mit der Initialisierung ist der PC nämlich auch gestartet worden, jedoch besteht noch keine Verbindung mit dem Anwender. Durch Starten des Programmes Mono oder Color (vom Kommando-Interpreter für Multitasking mit RUN) wird der PC aktiviert und man sieht eine besondere PC-Begrüßung, nämlich das Hydra-Bios (vollkompatibel, aber der Name ist doch gut gewählt). Dann erfolgt die normale Datums- und Zeitabfrage. Das PC-Fenster hat einen Rahmen, der nach zwei Maus-Tastendrücken verschwindet bzw. wieder auftaucht. Man gibt also brav Datum und Zeit ein und kann gleich arbeiten. Natürlich kann man auch die Fenstergröße ändern oder ein Amiga-Fenster aktivieren und dort irgend etwas machen. Bleiben wir aber beim PC: Das Fenster bietet bei voller Größe und ohne Rahmen genau 25 Zeilen mit je 80 Zeichen und MS-DOS (auch IBM-DOS 3.1) läuft tadellos. An dieser Stelle muß nicht weiter auf DOS-Operationen eingegangen werden, denn es funktioniert einfach alles genau wie beim PC-10/-20. Die Darstellung auf dem Amiga-Monitor ist einwandfrei, bei verkleinertem Fenster wird eben nur der entsprechende Ausschnitt gezeigt.

Anwendungen

Auf dem Sidecar lassen sich problemlos alle Programme starten, auch kopiergeschützte Disketten und Programme. Wordstar, Multiplan, dBase-II und -III, Lotus-1-2-3, Turbo-Pascal oder MS-Word mögen hier als Beispiele dienen, ebenso wie GW-Basic, Sidekick, Copy-

IIPC und die Norton-Utilities. Durch den identischen Aufbau entstehen eben keine Kompatibilitäts-Probleme.

Mit gewissen Einschränkungen muß man allerdings beim normalen Lieferumfang leben: Sidecar hat nur ein Laufwerk und 256 KByte RAM. Es besteht aber die Möglichkeit (9 freie Sockel), auf der Hauptplatine auf 512 KByte aufzurüsten. Extern kann auch ein zweites 360-KByte-Laufwerk angeschlossen werden (wahlweise 5¼ oder 3½ Zoll) – auch das Amiga Laufwerk 1010 läuft hier einwandfrei. Mit zwei Laufwerken und 512 KByte läßt sich angenehm arbeiten.

Auch IBM-Steckplätze (3 Stück) sind vorhanden. Wer also einen eigenen PC-Bildschirm haben möchte, kann hier eine Commodore AGA-Karte oder andere (Monochrom-, Herkuleskarte) einstekken. Selbstverständlich können auch Schnittstellen-Karten eingesteckt werden.

Am interessantesten dürfte hier die Möglichkeit sein, am Sidecar eine Festplatte anzuschließen, sei es eine „normale" extern oder, eleganter, eine Drivecard mit 20...30 MByte intern. Der Clou an dieser Festplatte ist nämlich die Möglichkeit, sie zwischen Amiga-DOS und PC-DOS aufzuteilen. Eine 30 MByte-Platte kann so z. B. zu 10 MByte dem PC gehören und die restlichen 20 MByte dem Amiga zur Verfügung stellen! Für alle die, die schnell eine Festplatte installieren wollen, ist Sidecar also eine gute Gelegenheit: Eine Amiga-Festplatte mit dazugehörigem Controller kostet zur Zeit etwa genausoviel wie Sidecar einschließlich 512 KByte RAM und einer Festplatten-Einsteckkarte – also volle Kapazität, Kompatibilität und kombinierte Amiga-PC-Fähigkeiten fürs gleiche Geld.

Am PC kennen viele Anwender das Problem, daß man zur vollen Ausnutzung einiger Programme entweder zwei Bildschirme (Farbe und monochrom) braucht oder eine Grafikkarte benötigt, die Farben zumindest als Grautöne darstellen kann. Mit solchen Halbheiten braucht man sich hier nicht abzugeben: Man startet einfach die beiden Tasks PC-Mono und PC-Color mit run, bedingt durch die DIP-Schalter-Stellung auf der PC-Platine befindet sich der Cursor beim Start auf dem Monochrom-Schirm. Mit dem MS-DOS-Befehl Mode kann man nun den Parameter CO80 übergeben, den Farbbildschirm anklicken und einfach dort weiterarbeiten. Da sich der Farbbildschirm wie jedes andere Amiga-Fenster verschieben läßt, bleiben auch das Monochrom- und die Amiga-Fenster weiterhin erreichbar. Der Wechsel erfolgt geradezu spielerisch einfach und dürfte auch jeden PC-Kenner überzeugen. Der Übergang von Farbe zum Monochrom-Bild erfolgt natürlich für den PC mit der Eingabe von MODE MONO im Farbfenster.

Funktion

Salopp formuliert laufen im Amiga zwei recht kleine Programme ab, die bei Aktivierung ihrer jeweiligen Fenster nichts anderes tun, als alle Tastatureingaben an den PC weiterzureichen. Die Schnittstelle weist keine Unterschiede zu einer normalen Tastatur auf, vielmehr sind hier PC und Amiga völlig selbstständig: Wer am PC das Programm Keybgr aufruft, hat eine deutsche Tastaturanordnung, wer aber vergißt, am Amiga SETMAP D aufzurufen, hat selber Schuld, wenn er dauernd Z und Y verwechselt. Beide Rechner sind auf Deutsch bzw. ASCII anpaßbar, wann Commodore allerdings die deutschen Tastaturkappen ausliefert, ist ungewiß. Man nehme kleine Etiketten, das alte Spiel...

Daß Sidecar noch weitere Bequemlichkeiten gegenüber dem PC bietet, sei hier auch nicht verschwiegen: Das sieht man am besten bei den Menüs, welche die PC-Fenster auf dem Amiga bieten: Das Project-Menü bietet die Optionen Save/Restore Settings, mit denen man einige Voreinstellungen (entsprechend den „Preferences") für den PC-Bereich abspeichern kann. Weiterhin gibt es den Punkt Info, der Informationen zu Fenstern, Farben und Tastatur zugänglich macht, und den Punkt Close, um aus dem PC-Modus auszusteigen (allerdings wird der PC durch erneutes Starten von Mono oder Color an der gleichen Stelle wieder aktiviert).

Das besagte Display-Menü behandelt zuerst Size und Border, womit das Fenster erstmal schnell klein (so wie man zuletzt das PC-Fenster selbst verkleinert hat) oder groß (1:1) und der Fensterrahmen ein- oder abgeschaltet wird. Das Color-Menü ist faszinierend, besonders auf dem Farbbildschirm: Manche PC-Programme haben ja geradezu schmerzhafte Farbeinstellungen, die nicht oder nur mit großem Hackaufwand änderbar sind. Wie bei Preferences oder Graphicraft ist auf dem Amiga jede dargestellte Farbe frei änderbar, bis zu vier (die Workbenchfarben) im Monochrom- und bis zu 16 (PC-Farben) im Farbmodus. Als nächstes ist die Cursor-Blinkrate einstellbar, von ruhig bis zum Infarkttempo. Auch die Option „Open another window" ist praktisch: Wer eine Textverarbeitung ohne Fenster hat, kann im Amiga-Speicher eine Kopie des PC-Fensters bereithalten, ein neues Fenster öffnen, dort weiterarbeiten und dabei doch einen ganz anderen Abschnitt seines Textes einsehen. Sicher fällt jedem Anwender dazu noch mehr ein. Die Funktion „Refresh display" baut nach diversen Fenstergrößen-Änderungen den Inhalt korrekt neu auf, allerdings trat hier nie ein Fall auf, in dem das nicht von alleine gut gegangen wäre. Mit „Text display depth" kann man dem Amiga die Arbeit etwas erleichtern, indem man ihm nicht die Darstellung von 16, sondern nur acht oder vier Farben abverlangt. Bei intensivem Multitasking kann das von Nutzen sein, ebenso wie die Einstellbarkeit der Display-Priorität in Fünferschritten, die festlegt, wie intensiv sich der Amiga um den Aufbau des PC-Fensters kümmern soll. Zu guter letzt kann man noch das Zeilensprung-Verfahren (Interlace) ein- oder abschalten, aber dazu braucht man einen sehr viel besseren Monitor...

Kleine Mängel

In der ersten Testversion hatte die Sidecar-Software für den Amiga zumindest drei Fehler, die aber in der neuen Softwareversion WorkBench 1.2 Sidecar+ behoben sind. Die Sonderfunktionen NumLock, ScrollLock und PrtSc führten zu einer Unterbrechung der Verbindung Amiga-Sidecar, nach Neustart mit PC mono bzw. PC color konnte jedoch ohne Datenverlust weitergearbeitet werden.

Die Alt-Ziffernblockeingabe beliebiger Characters funktionierte auch nicht einwandfrei. Schwerwiegendster Fehler war die falsche Darstellung hochauflösender Graphik in Form von vier kleinen statt eines normalen Bildes, einfach nachzuweisen mit dem Basic-Befehl Screen 2. Alle diese Fehler sind jedoch mit Verwendung der neuen Software behoben. Sidecarbesitzer, die immer noch diese Probleme haben, können bei ihren Händlern die Softwareupdates erhalten. Lediglich bei älteren Modellen des Sidecars mit alter Systemplatine muß zusätzlich ein Eingriff von einem Techniker vorgenommen werden, da die neue Software sonst sofort abstürzt. Mit der erfolgten schnellen Behebung der Fehler hat Sidecar eine echte Chance, zumindest wage ich die Behauptung, daß es keine Hardware-Probleme gibt.

Was Commodore noch verspricht: Programme zur Datei-Übertragung, so daß der Amiga die PC-Disketten liest und umgekehrt. Dann kann endlich jeder PC-Benutzer, der eine leistungsfähige Rechnerumgebung braucht, seine Arbeit nach und nach auf den Amiga verlagern. Eine Speichererweiterung, die für den Amiga gemacht ist, jedoch im Sidecar untergebracht werden soll, ist auch bei Commodore in der Planung, aber da gibt es ja notfalls auch einige andere Anbieter.

Lieferumfang

Mit dem Sidecar erhält man ein Commodore-typisch dünnes, aber sehr gutes Handbuch, zwei 3½-Zoll-Disketten mit dem Sidecar-System 1.2, Workbench und Kickstart 1.2 sowie eine 5-¼-Zoll-Diskette mit MS-DOS 2.11 für PC-10/20 und das dazugehörige DOS-Handbuch. Benötigt wird ein Amiga mit mindestens 512 KByte RAM und einem Disketten-Laufwerk, empfehlenswert sind aber ein paar MByte RAM (wie immer beim Amiga, mit 512 KByte läuft bei aktivem Mono- und Color-Fenster nur noch ein CLI-Fenster und sonst nichts). Auch ein zweites Diskettenlaufwerk schadet beim Amiga nie, bei einer Festplatte im Sidecar ist das allerdings nebensächlich. Sidecar sollte man immer auf 512 KByte aufrüsten; wer keine Festplatte hat, sollte zumindest eine RAM-Disk einrichten.

Ein wesentlicher Kritikpunkt bei Sidecar ist noch der Lüfter: Er lärmt nämlich für zwei, mit für den Amiga-Lüfter, den man nicht hört. Hier könnte und sollte im Interesse der Nerven aller Benutzer bald Abhilfe geschaffen werden.

Ansonsten ist Sidecar ideal für Anwender, die auf ein Hochleistungs-System umsteigen wollen, ohne Ihre PC-Software wegzuwerfen. Sobald die Datei-Übertragung läuft, wird dieser Umstieg wirklich einfach. Eine weitere Zielgruppe sind Leute, die ihren Amiga professionell nutzen wollen, eine Platte brauchen und diese nicht allein für rund 4500 DM kaufen wollen. Man muß zwar für Sidecar nur 1650 DM zahlen, erhält aber eben wesentlich mehr für sein Geld.

Transputer konkret

Die Firma Parsytec, Aachen, hat sich darauf spezialisiert, neue Technologien der Informationsverarbeitung für die industrielle Anwendung nutzbar zu machen. Als weltweit erste Firma hat Parsytec ein Transputersystem vorgestellt, das zum Einsatz in der Industrie vorgesehen ist. Dieses System heißt Megaframe. Die Konzeption beruht auf der Vernetzbarkeit der Transputer-Chips. Zum Beispiel bietet die Baugruppe VTF (Versatile Transputer Frontend dem Systementwickler drei Transputereinheiten auf einer Europakarte. Über 30 MIPs sind damit erreichbar.

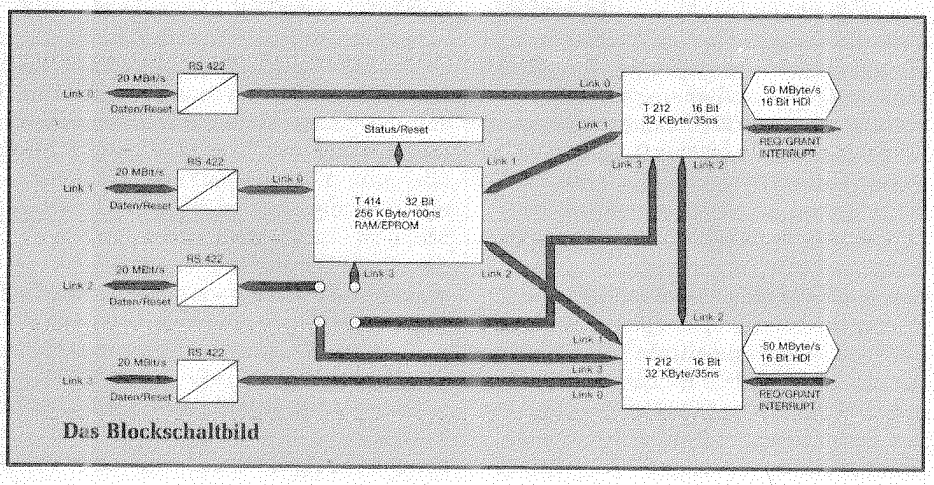
Von den drei unabhängigen Prozessorknoten des VTF enthält der erste einen 32-Bit-T414-Transputer mit 256 KByte statischem RAM. Prozessortakt und Speicherzugriff sind 17 MHz und 120 ns, optional 20 MHz und 100 ns. Anwendungsprogramme lassen sich von außen dynamisch über Kommunikationskanäle (links) laden. In Standalone-Anwendungen werden 128 KByte durch EPROM mit Programmcode für alle drei Prozessoren ersetzt. Durch extrem kompakten Tansputercode und Realtime-Scheduling in Hardware entspricht dies ca. 256 KByte Code konventioneller Prozessoren.

Zwei weitere 16-Bit-Knoten enthalten jeweils einen T212-Transputer mit 32 KByte statischem RAM. Prozessortakt und Speicherzugriff sind 17 MHz und 45 ns, optional 20 oder 22 MHz und 35 ns. Bei voller Softwarekompatibilität mit dem T414 erreichen die 16-Bit-Transputer mit multiplexfreiem Datenbus durchschnittliche Befehlsausführungszeiten zwischen 45 und 135 ns. Über High Speed Data Interfaces (HDI-Flachkabelanschluß nach außen) greift jeder T212 auf ein Adreßfenster von weiteren 32 KByte außerhalb des Board zu. Die Datentransfer-Rate beträgt programmgesteuert 8,7 MByte/s, optional 10 bzw. 11,4 MByte/s, bei externem DMA-Controller bis 40 bzw. 50 MByte/s. Das HDI-Interface zu Custom Hardware ist flexibel und einfach. EXTREQUEST/EXTGRANT synchronisieren den Zugriff. EXBYTE schaltet auf 8 Bit Zugriffsbreite um. EVENT/EVENTACK steuern Interrupts mit typisch 0,95 (max. 2,6) Mikrosekunden Reaktionszeit bis zum ersten Befehl.

Die zwölf Transputer-Links des VTF arbeiten mit 20 MBit/s. Sie verbinden standardmäßig die Prozessoren untereinander und mit der Außenwelt. Drei von ihnen sind mit Jumpern konfigurierbar für zusätzliche Verbindungen untereinander oder zur Außenwelt. Die Außenverbindungen sind gemäß RS-422 gepuffert und nach dem MEGAFRAME-Standard auf die VG-Leiste geführt. Mit Standardkabel werden bei 20 MBit/s 10 m Reichweite garantiert; größere Entfernungen sind durch Jumper für 10 bzw. 5 MBit/s möglich.

Parallel zu den Links ist bidirektional ein programmgesteuerter Reset implementiert. Damit kontrolliert in einem MEGAFRAME-Netzwerk jeder Prozessor über die Kommunikationsverbindungen die Aktivitäten der vier nächsten Nachbarn. Im Fehlerfalle setzt er sie zurück und lädt sie über die gleichen Links dynamisch mit neuem Programmcode. Ein Statusregister hält Fehlerzustände des T414 sowie Adreßfehler aller drei Transputer fest. Im Fehlerfall eines 16-Bit-Knoten wird dieser vom T414 zurückgesetzt und neu gestartet. Das VTF-Modul im extended Single-Euroformat (100 mm × 220 mm) hat eine Stromaufnahme unter 3 A bei 5 V.

Nach Unterlagen von Parsytec.



Das Blockschaltbild

Helmut Assig, Walter L. Giesler

D/A-Umsetzer für den PC

Professioneller D/A-Umsetzer mit 12 Bit Auflösung

D/A-Umsetzer-Baugruppen für den professionellen Einsatz sollten genau sein und gleichzeitig den Anwender zwischen Spannungs- und Stromausgang (Prozeßstrom 4...20 mA) wählen lassen. Die mc-D/A-Umsetzer-Karte kann mit bis zu vier unabhängigen D/A-Kanälen bestückt werden.

Immer häufiger werden PCs in technischen Anwendungen eingesetzt. Sie werten z. B. Meßreihen aus, unterstützen Entwicklungsabteilungen beim Entwerfen neuer Produkte und steuern technische Prozesse. Als Bindeglied zur Außenwelt benötigt man neben digitalen Ein- und Ausgängen, die z. B. Schalter abfragen oder Magnetventile betätigen, auch Koppler für analoge Signale. Eingangsseitig sind dies A/D-Umsetzer, die analoge (sich stetig ändernde) Signale in digitale Werte umsetzen. Als D/A-Umsetzer werden Einheiten bezeichnet, die einen digitalen Wert in ein proportionales Signal umsetzen. Das analoge Signal kann eine Spannung oder ein Strom sein. Entsprechend den speziellen Anforderungen an D/A-Umsetzer für den professionellen Einsatz ist die hier vorgestellte Baugruppe mit D/A-Umsetzern hoher Geschwindigkeit und hoher Genauigkeit bestückt. Das Analogsignal kann als Spannung in den Bereichen 0...5 V, 0...10 V, −2,5 V...+2,5 V, −5 V...+5 V und −10 V...+10 V abgegeben werden. Mit einer Zusatzbestückung kann ein Strom von 4...20 mA abgegeben werden, wobei 4 mA der Nullstellung entsprechen. Bei dieser als Prozeßstrom bezeichneten Übertragung kann zwischen Nullstellung und Leitungsbruch unterschieden werden, da bei Leitungsbruch kein Strom fließt, in der Nullstellung jedoch der 4-mA-Minimalstrom.

Der Spannungsausgang von ±10 V wird häufig zur Steuerung von Servoverstärkern für Gleichstrommotoren eingesetzt. Zusammen mit einer Einlesebaugruppe für Positionsgeber (Inkremental- oder Winkelschrittgeber) sind Positionsregelungen möglich. Da die Baugruppe eine hohe Umsetzgeschwindigkeit (max. 4 µs für einen Spannungssprung über den ganzen Bereich) hat, kann sie für alle Arten von digitalen Regelungen eingesetzt werden. Selbst für eine Vielzahl von digitalen Filtern wird die Karte einsetzbar sein. Das begrenzende Element ist lediglich die Rechengeschwindigkeit des Prozessors.

Das erste Auswahlkriterium eines D/A-Umsetzers ist die Auflösung, welche meistens in Bit angegeben wird. Mit 8 Bit lassen sich 256 verschiedene Werte darstellen, mit 12 Bit 4096 (2 hoch 12 = 4096). Eine 12-Bit-Auflösung bedeutet bei einem D/A-Wandler, daß sich das maximale Ausgangssignal in 4096 Schritte unterteilt. Um mit einem D/A-Umsetzer genau arbeiten zu können, ist es nicht nur erforderlich, daß das Signal in 4096 Schritte (bei 12 Bit) unterteilt wird, sondern auch noch, daß alle Schritte möglichst gleich groß sind. Die Abweichungen vom rechnerischen Wert werden Nichtlinearität genannt. Für die hier vorgestellte Baugruppe beträgt die maximale Nichtlinearität ±½ LSB, d. h., ± die Hälfte eines Schrittes.

Da die Baugruppe DA12 mit 12-Bit-Umsetzern bestückt ist, bedeutet das weniger als 0,025 % (1:4096). Um diese sehr hohe Genauigkeit des D/A-Bausteins auch wirklich am Ausgang der Baugruppe verfügbar zu haben, müssen das Layout und die Beschaltung sorgfältig durchgeführt werden. Ein Operationsverstärker im Signalzweig sollte mit Widerständen beschaltet werden, die genauer als 0,025 % sind. Dies wäre ein schon fast unlösbares Beschaffungsproblem. Auf der Baugruppe werden daher Wandlerbausteine eingesetzt, die die Re-

Tabelle 1: Der E/A-Adreßbereich des IBM-PC

E/A-Adresse	A9	Funktion	E/A-Adresse	A9	Funktion
000H...00FH	0	DMA-Controller (8237A-5)	2F8H...2FFH	1	Zweite serielle Schnittstelle
020H...021H	0	Interrupt-Controller (8259-5)	300H...31FH	1	Prototypkarte
040H...043H	0	Zeitgeber/Zähler (8253-5)	320H...32FH	1	Festplatten-Controller
060H...063H	0	Systemregister (8255A-5)	378H...37FH	1	Druckerschnittstelle (parallel)
080H...083H	0	DMA-Seitenregister (74LS670)	380H...38FH	1	SDLC-Schnittstelle
0AXH	0	NMI-Interrupt-Register	3A0H...3AFH	1	Reserviert
0CXH	0	Reserviert	3B0H...3BFH	1	Monochromadapter und Drucker
0EXH	0	Reserviert	3C0H...3CFH	1	Reserviert
100H...1FFH	1	Nicht verwendet	3D0H...3DFH	1	Farbgrafikkarte
200H...20FH	1	Für Computerspiele (Game Port)	3E0H...3E7H	1	Reserviert
210H...217H	1	Erweiterungseinheit	3F0H...3F7H	1	Floppy-Controller
220H...24FH	1	Reserviert	3F8H...3FFH	1	Serielle Schnittstelle
278H...27FH	1	Zweiter Drucker			

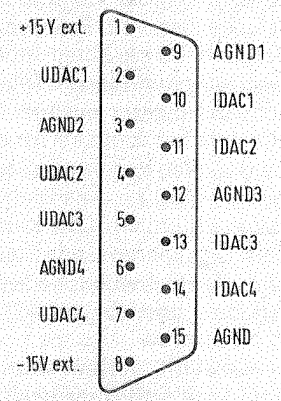


Bild 1. Anschlußbelegung der Ausgangsbuchse ST1

ferenzspannungsquelle, den Umsetzer und den Strom-Spannungswandler im Ausgangszweig komplett integriert haben.

Die Baugruppe kann im PC, XT und AT verwendet werden. Sie belegt acht Adressen, die im 1024 Byte großen E/A-Adreßbereich (*Tabelle 1*) des PCs frei einstellbar sind. In aufsteigender Reihenfolge werden die vier DACs (DAC = Digital/Analog Converter) mit Low- und High-Byte angesprochen (*Tabelle 2*). Da der PC-Bus nur ±12 V als Hilfsspannung zur Verfügung stellt, für Ausgangsspannungsbereiche bis 10 V jedoch ±15 V erforderlich sind, kann diese Hilfsspannung entweder von außen zugeführt oder über ein Spannungswandlermodul aus den +12 V des PCs erzeugt werden.

Die Anschlüsse der D/A-Umsetzer sind auf eine 15polige D-Sub-Buchse geführt (*Bild 1*). Pro Kanal sind die drei Signale Analogmasse, Spannungsausgang und der optionale Stromausgang verfügbar.

Schaltungsbeschreibung

Der PC-Bus besteht aus 8 Datenleitungen, 20 Adreßleitungen, den Prozessor-Steuersignalen und Hilfssignalen für Interrupt und DMA (*Bild 2*).

Die Karte verwendet, wie bei E/A-Karten üblich, 10 Adreßleitungen (A0...A9), 8 Datenleitungen, IOR, IOW, AEN und die Betriebsspannungen +5 V und ±12 V. Zur Adreßdecodierung werden A3...A9 auf zwei Vergleicherbausteine (74LS85) geführt. Diese vergleichen die Bus-Adresse mit der an den Steckbrücken des Feldes J10 eingestellten Baugruppenadresse. *Tabelle 3* zeigt die Adreßbit-Zuordnung des Steckbrückenfelds. Die Einstellung der freien Adressen im Bereich 250H...31FH ist aus *Tabelle 4* ersichtlich.

Mit den Signalen A0...A2 wird die Feindecodierung der acht Adressen durchgeführt. Abhängig vom Zustand von A1 und A2 wird einer der vier D/A-Wandler ausgewählt und mit $\overline{CS}$ aktiviert. Um nur bei Ausgabevorgängen die D/A-Umsetzer anzusprechen, wird der Decoder (74LS138) erst bei $\overline{IOW}$ = 0 (I/O-Write) freigegeben. Gleichzeitig wird überprüft, ob es sich um einen CPU-Zyklus oder um einen DMA-Zyklus handelt (AEN). Aus den Auswahlsignalen wird auch das Freigabe-Signal ($\overline{G}$) des Datenbustreibers 74LS245 erzeugt.

Durch den Datenbustreiber wird der Bus geringer belastet – der Eingangsstrom (High) wird von 4 × 300 µA auf 20 µA verringert – und die D/A-Umsetzer werden vom Bus entkoppelt. D/A-Umsetzer, insbesondere monolithische, sind anfällig gegen Änderungen des digitalen Eingangssignals. Digitale Durchgriffe auf das Ausgangssignal werden durch den Datenbustreiber verhindert. Die Adresse A0 geht über Treiber direkt auf die vier D/A-Umsetzer, um zwischen unteren und oberen 8 Bit auszuwählen. Im Schaltplan ist nur ein D/A-Kanal dargestellt, da die anderen Kanäle gleich aufgebaut sind (*Bild 3*).

Tabelle 2: Adreßbelegung der Baugruppe

Offset zur Basis-Adresse	Register
0	DAC1 Low
1	DAC1 High
2	DAC2 Low
3	DAC2 High
4	DAC3 Low
5	DAC3 High
6	DAC4 Low
7	DAC4 High

Low = niederwertiges Byte D0...D3
High = höherwertiges Byte D4...D11

Tabelle 3: Adreßbit-Zuordnung des Steckbrückenfelds J10

Adreßleitung	Position
A3	A
A4	B
A5	C
A6	D
A7	E
A8	F
A9	G

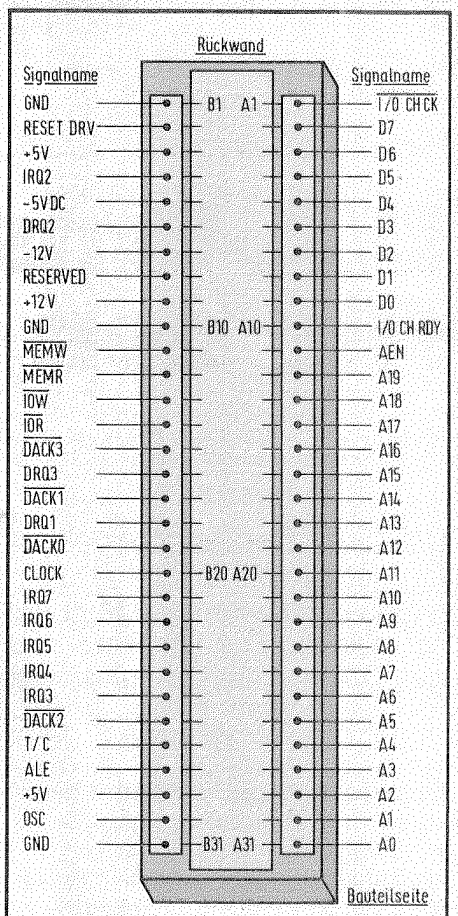


Bild 2. PC-Bus-Anschlußbelegung

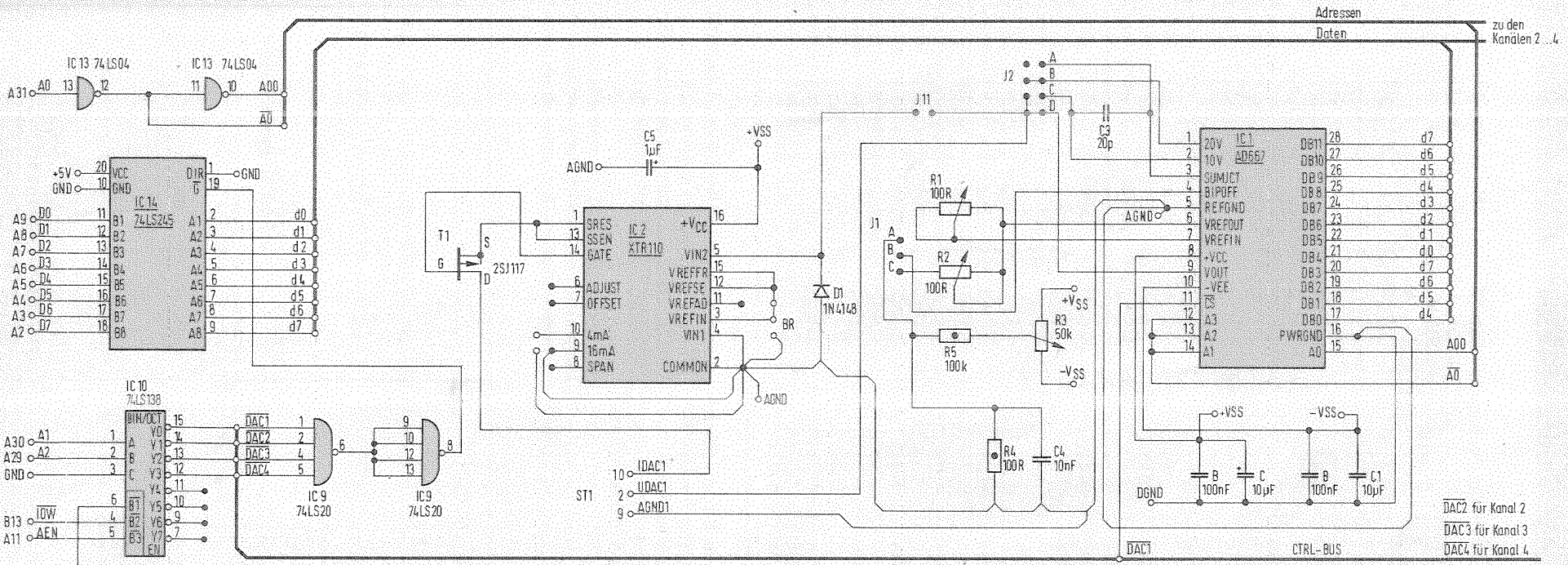
Die D/A-Umsetzer

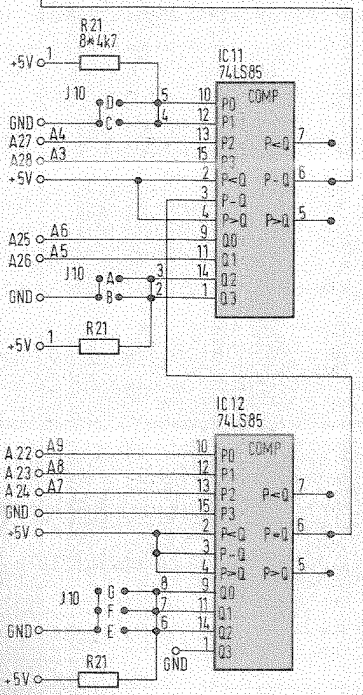
Der auf der Baugruppe DA12 eingesetzte D/A-Umsetzer AD667 ist der höchstintegrierteste monolithische 12-Bit-D/A-Umsetzer. Er besteht aus drei 4-Bit-Vorregistern, einem 12-Bit-Halteregister, der Referenzspannungserzeugung und dem Ausgangsverstärker für den Spannungsausgang (*Bild 4*).

Da der PC-Bus nur mit acht Datenleitungen ausgestattet ist, muß der D/A-Umsetzer in zwei Schritten geladen werden.

Dies ist beim AD667 möglich, da dieser über drei 4-Bit-Vorspeicher verfügt.

Über drei Auswahlleitungen (A0...A2), mit A0 für das niedrigstwertigste Nibble (Nibble = ½ Byte = 4 Bit) und A2 für das höchstwertigste, werden die 4-Bit-Vorspeicher angesprochen. Bei A3 = 0 werden die Werte aus den Vorspeichern in den Haltespeicher übernommen . Zunächst werden die unteren 4 Bit aus den Busleitungen D4...D7 geladen. Dazu werden A0 auf logisch 0 und A1...A3 auf logisch 1 gesetzt. Im zweiten Buszyklus





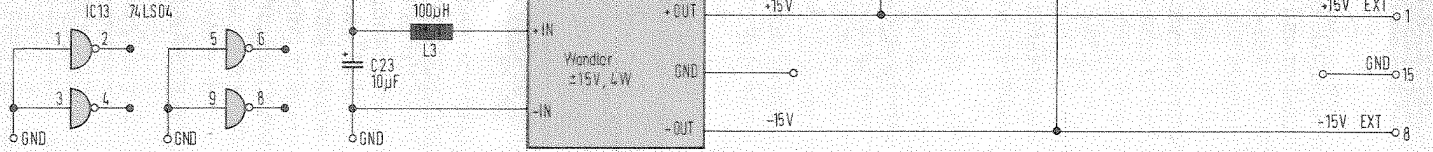


Bild 3. Das Schaltbild

werden die acht höherwertigen Daten angelegt und transparent durchgeschaltet. Dies geschieht mit A0 = 1, da das unterste Nibble bereits geladen ist und mit A1...A3 = 0. A1 und A2 wählen die oberen beiden Register aus und bei A3 = 0 schalten alle Register in den Haltespeicher durch. Intern wird aus dem digitalen Code ein proportionaler Strom erzeugt, der in einem schnellen Verstärker in eine Ausgangsspannung umgesetzt wird.

Die Spannungsumsetzungsstufe ist ein als I/U-Wandler geschalteter Operationsverstärker, der aus einem negativen Strom eine positive Spannung erzeugt.

In einer I/U-Schaltung läßt sich der Ausgangsspannungsbereich durch einen Rückkopplungswiderstand einstellen.

Für verschiedene Bereiche sind zwei 5-kΩ-Widerstände bereits fest eingebaut. Auf dem AD667 wird die notwendige hohe Genauigkeit durch Laserabgleich erreicht. *Bild 5* zeigt, wie sich die Ausgangsspannungsbereiche mit Steckbrükken einstellen lassen.

Die verschiedenen maximalen Ausgangsspannungen ergeben sich durch Parallelschaltung, Kurzschluß oder Serienschaltung der beiden 5-kΩ-Widerstände. Damit läßt sich der Ausgangsspannungsbereich einstellen, nicht jedoch die Polarität. Bislang wird aus dem negativen Strom von 0...−2 mA eine positive Ausgangsspannung zwischen 0 und dem Maximalwert erzeugt. Um eine negative Ausgangsspannung zu bekommen, müßte ein positiver Strom vorhanden sein. Mit dem Eingang Bipolar-Offset kann eine negative Ausgangsspannung erzeugt werden. Für den unipolaren Betrieb wird dieser Eingang auf die virtuelle Masse gelegt. Mit einem Trimmer (R3, R8, R13, R18) wird der Nullpunkt eingestellt.

Für den bipolaren Betrieb wird die Referenzspannung von 10 V (Pin 4) über einen 100-Ω-Trimmer auf den Eingang Bipolar-Offset (Pin 6) gegeben.

Über den 9,95-kΩ-Widerstand läßt sich am invertierenden Eingang des Ausgangsverstärkers ein Strom von genau

Tabelle 4: Einstellbeispiele für die Basisadresse

Port adresse	A A3	B A4	C A5	D A6	E A7	F A8	G A9
1000	-	-	-	-	-	offen	-
1080	offen	-	-	-	-	offen	-
1100	-	offen	-	offen	-	-	offen
1140	offen	offen	-	offen	-	-	offen
1200	-	-	offen	offen	-	-	offen
1240	offen	-	offen	offen	-	-	offen
1300	-	offen	offen	offen	-	-	offen
1340	offen	offen	offen	offen	-	-	offen
1400	-	-	-	-	offen	-	offen
1440	offen	-	-	-	offen	-	offen
1500	-	offen	-	-	offen	-	offen
1540	offen	offen	-	-	offen	-	offen
1600	-	-	offen	-	offen	-	offen
1640	offen	-	offen	-	offen	-	offen
1700	-	offen	offen	-	offen	-	offen
1740	offen	offen	offen	-	offen	-	offen
1800	-	-	-	offen	offen	-	offen
1840	offen	-	-	offen	offen	-	offen
1900	-	offen	-	offen	offen	-	offen
1940	offen	offen	-	offen	offen	-	offen
1A00	-	-	offen	offen	offen	-	offen
1A40	offen	-	offen	offen	offen	-	offen
1B00	-	offen	offen	offen	offen	-	offen
1B40	offen	offen	offen	offen	offen	-	offen
1C00	-	-	-	offen	offen	offen	offen
1C40	offen	-	-	offen	offen	offen	offen
1D00	-	offen	-	offen	offen	offen	offen
1D40	offen	offen	-	offen	offen	offen	offen
1E00	-	-	offen	offen	offen	offen	offen
1E40	offen	-	offen	offen	offen	offen	offen
1F00	-	offen	offen	offen	offen	offen	offen
1F40	offen	offen	offen	offen	offen	-	offen
2000	-	-	-	-	-	offen	offen
2040	offen	-	-	-	-	offen	offen
2100	-	offen	-	-	-	offen	offen
2140	offen	offen	-	-	-	offen	offen

- = Brücke gesteckt

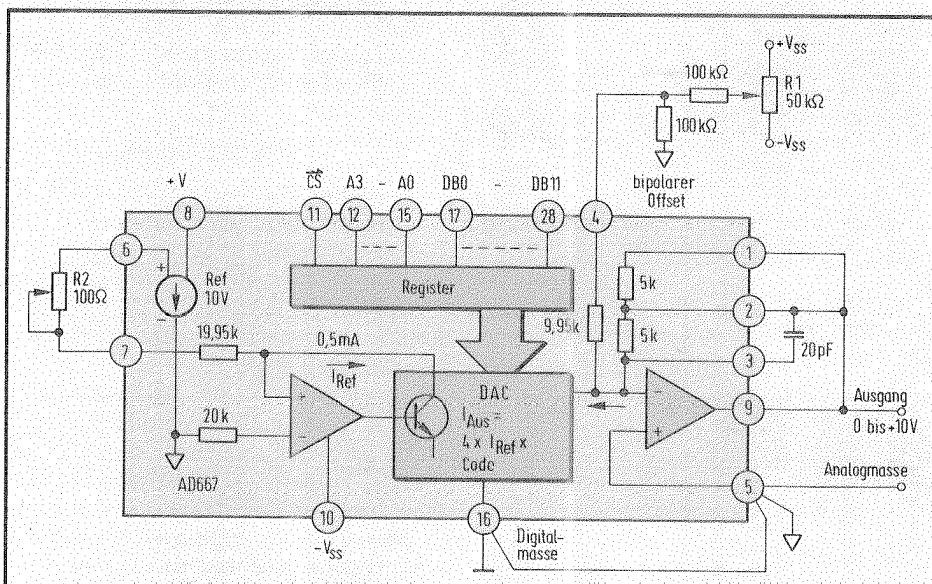


Bild 4. Funktionsprinzip des Analogteils

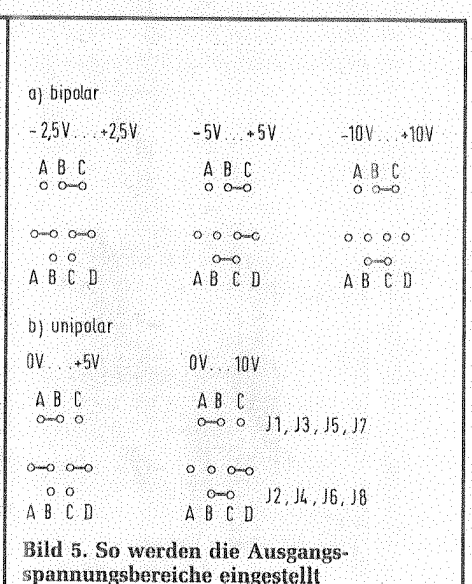


Bild 5. So werden die Ausgangsspannungsbereiche eingestellt

1 mA addieren. Der Widerstand hat den Wert 9,95 kΩ, damit sich zusammen mit dem 100-Ω-Trimmpoti ein Abgleichbereich von ±50 Ω ergibt. Der resultierende Strombereich am Eingang des Operationsverstärkers ist +1 mA...−1 mA. Der Ausgangsspannungsbereich bleibt erhalten, jedoch um 50% nach unten versetzt. Entsprechend wird aus dem unipolaren Ausgangsspannungsbereich 0...+10 V der bipolare Bereich −5...+5 V. Mit jeweils drei Trimmpotis lassen sich die D/A-Umsetzer abgleichen (*Tabelle 5*).

Um die Referenzspannung abgleichen zu können, wird der größte binäre Wert (12 × „1") linksbündig ausgegeben. Im unipolaren Betrieb wird der Ausgang auf 10 V minus ein LSB eingestellt (9,9976 V). Beim bipolaren Betrieb entsprechen 10 V minus einem LSB 9,9952 V. Zum Offsetabgleich muß auf den D/A-Umsetzer Null ausgegeben werden, wobei im unipolaren Betrieb der Ausgang auf Null und im bipolaren Betrieb auf die maximale negative Spannung abgeglichen wird.

Der Prozeßstrom-Umsetzer

In professionellen Anwendungen werden Stromausgänge mit 4...20 mA noch häufiger verwendet als Spannungsausgänge. Unsere Baugruppe kann daher mit speziellen Wandlerbausteinen, die Spannung in Prozeßstrom umsetzen, erweitert werden. Der Umsetzer ist vollständig in einem IC untergebracht und braucht, um Wärmedrift zu vermeiden, einen Ausgangstransistor. Die integrierte Schaltung mit der Bezeichnung XTR110 (*Bild 6*) benötigt eine Versorgungsspannung, die größer als 13,5 V ist, so daß entweder ±15 V extern angeschlossen werden müssen oder die Karte mit einem Gleichspannungswandler bestückt werden muß. Als Spannungsbereich am Eingang des Prozeßstrom-Umsetzers (V_{in2}) werden 0...5 V gewählt. Eine

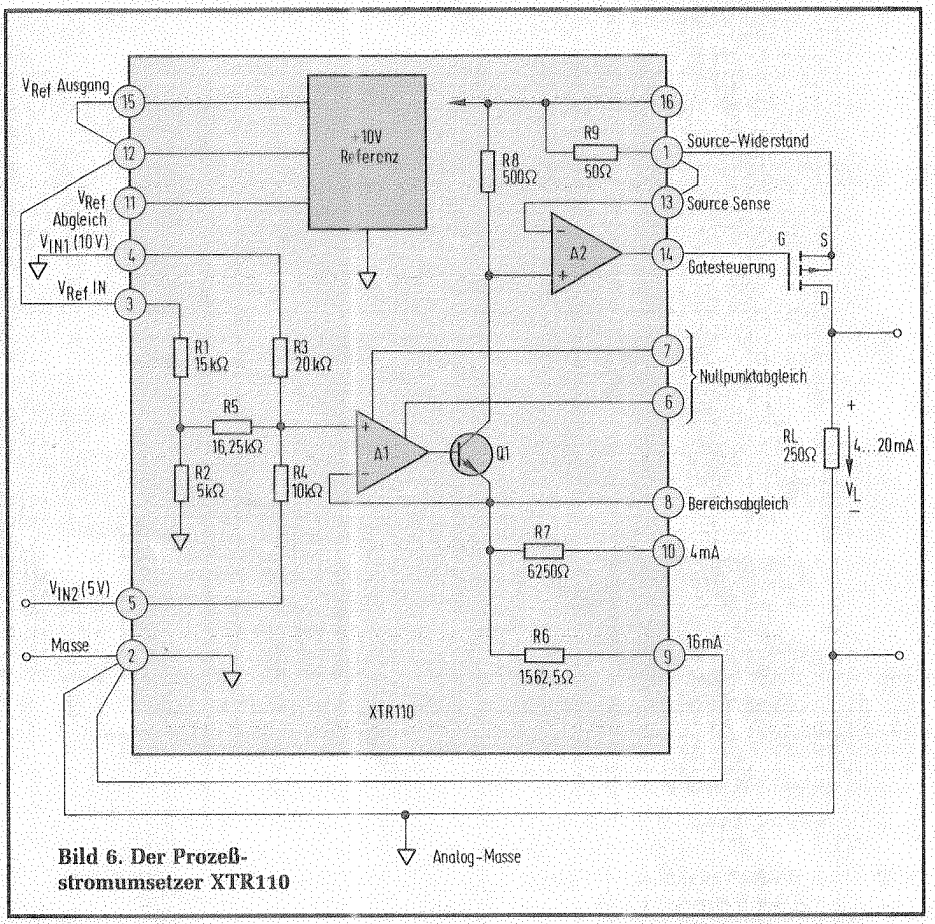


Bild 6. Der Prozeßstromumsetzer XTR110

Schutzdiode zwischen dem Eingang V_{in2} und der Masse des Analogteils schützt den XTR110 vor negativen Eingangspannungen. Der im XTR110 integrierte Operationsverstärker A1 setzt mit dem externen Transistor Q1 und dem Widerstand R6 Eingangsspannungen im Bereich von 0...5 V in Ausgangsströme von 0...16 mA

Tabelle 5: Einstellelemente

	DAC1	DAC2	DAC3	DAC4
Offset-Abgleich unipolarer Betrieb	R3	R8	R13	R18
Offset Abgleich bipolarer Betrieb	R2	R7	R12	R17
Referenzabgleich	R1	R6	R11	R16
Unipolar/bipolar Umschaltung	J1	J3	J5	J7
Ausgangsspannungsbereich	J2	J4	J6	J8
U/I-Wandler	J11	J12	J13	J14

Tabelle 6: Die Stückliste

D1...D4	1N4148
T1...T4	2SJ117 (Hitachi)
IC1, IC3, IC5, IC7	AD667 (Analog Devices)
IC2, IC4, IC6, IC8	XTR110 (Burr Brown)
IC9	74LS20
IC10	74LS138
IC11, IC12	74LS85
IC13	74LS04
IC14	74LS245
R1, R2, R6, R7, R11	100 Ω Cermet-Trimmer
R12, R16, R17	100 Ω Cermet-Trimmer
R3, R8, R13, R18	50 kΩ Cermet-Trimmer
R4, R9, R14, R19	100 Ω
R5, R10, R15, R20	100 kΩ
R21	Widerstandsnetzwerk 8 × 4,7 kΩ
C1, C2, C7, C8, C11, C12	10 µF Tantal
C16, C17, C23, C26	10 µF Tantal
C3, C8, C13, C18	20 pF Keramik
C4, C9, C14, C19	10 nF Keramik
C5, C10, C15, C20, C21, C22	1 µF Tantal
B (16 Stück)	100 nF Keramik
L1...L3	100 µH
Spannungswandler 12/1515/6 W	(optional)
15polige D-Sub-Buchse	

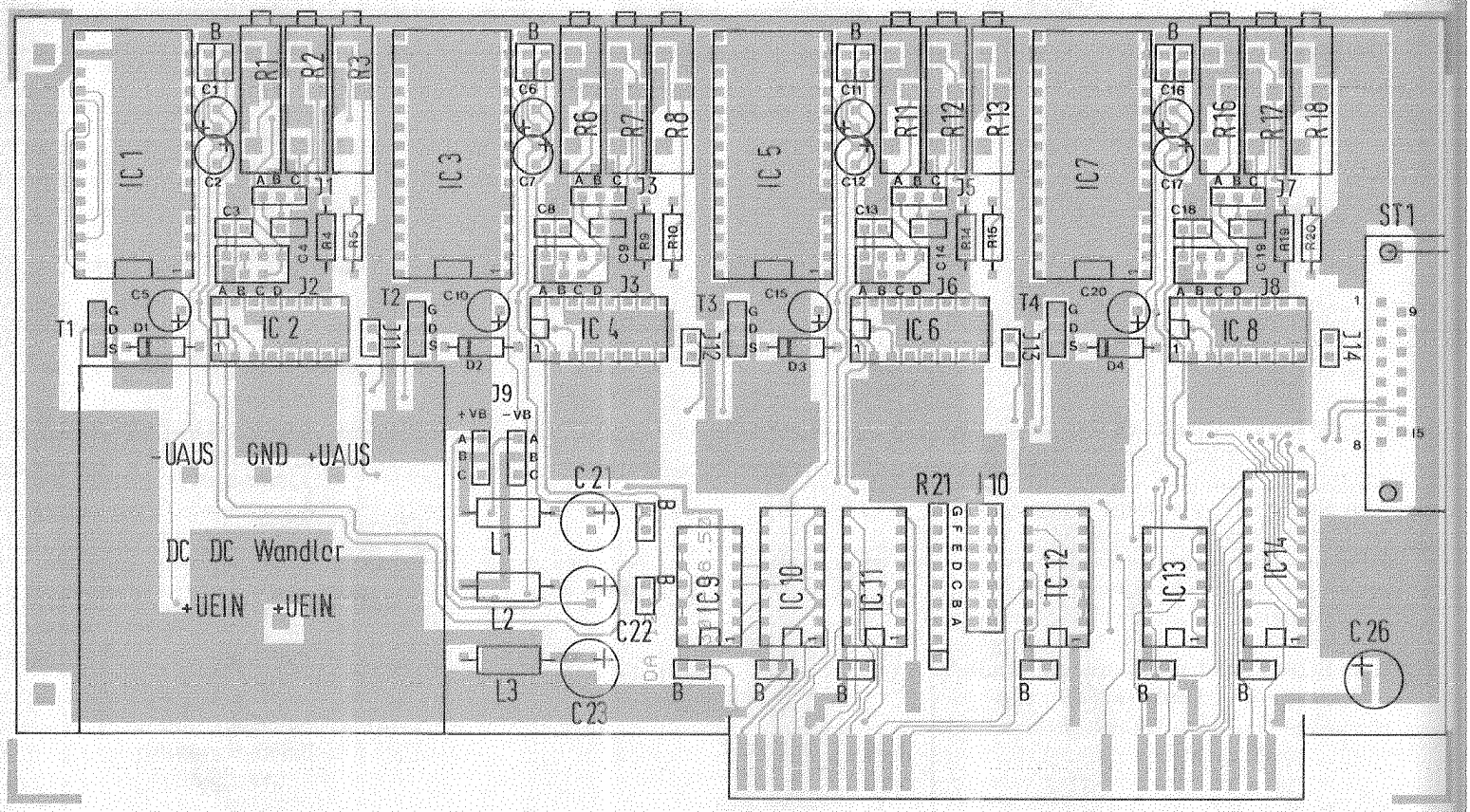


Bild 7. Der Bestückungsplan

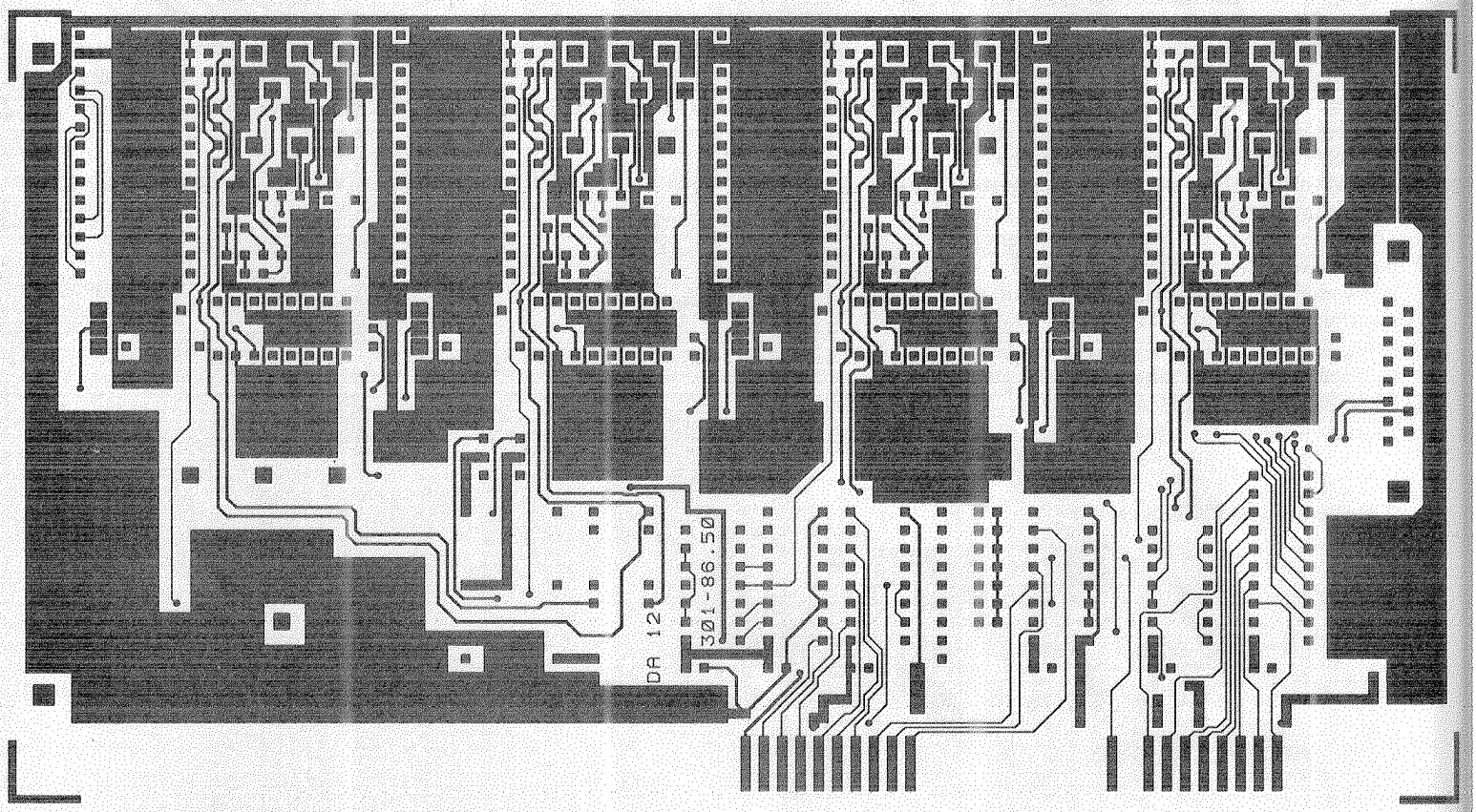


Bild 8. Die Bestückungsseite

um. Zu der im XTR110 integrierten 10-V-Referenz wird ein Teil der Eingangsspannung addiert. Dieser Offset ist genauso groß, daß der Ausgangsstrom um 4 mA erhöht wird und sich ein Ausgangsstrombereich von 4...20 mA ergibt.

Intern arbeitet der XTR110 nur mit einem Zehntel des Stroms. Über den integrierten Widerstand R9 zwingt der Strom-/Stromumsetzer A2 den externen Transistor T, genau den zehnfachen Strom abzugeben. Auch beim XTR110 sind die Widerstände integriert, um die erforderliche Genauigkeit zu erreichen.

Der externe Leistungstransistor (2SJ117) ist ein P-Kanal-MOSFET im TO-220-Gehäuse. Neben dem Ausgangsstrombereich von 0...16 mA läßt sich der XTR110 auch mit 0...20 mA und 5 bis 25 mA betreiben. Auf der Lötseite der Karte können diese Bereiche mit Lötbrücken eingestellt werden. Da die Ausgangsstrombereiche 0...20 mA und 5...25 mA selten benötigt werden, ist die Karte für den Bereich 4...20 mA fest verdrahtet.

Wird einer der beiden anderen Bereiche benötigt, so muß diese Verbindung aufgekratzt werden.

Spannungsversorgung

Da bei der hohen Genauigkeit der Baugruppe eine saubere Betriebsspannung unabdingbar ist, werden die Hilfsspannungen +VSS und −VSS über LC-Filter gesiebt. Die Filtereingänge können über J9AB mit den ±12 V des PC-Busses oder über BC mit den externen ±15 V bzw. mit dem internen Spannungswandler verbunden werden. Der interne Wandler wird aus den +12 V des PC-Busses versorgt, wobei ein weiteres LC-Filter verhindert, daß die +12 V durch den Schaltwandler verunreinigt werden.

Wenn die Baugruppe mit einem Wandler bestückt ist, dann sind am Ausgangsstecker die Hilfsspannungen verfügbar.

Auf keinen Fall dürfen über den Ausgangsstecker externe ±15 V bei arbeitendem Wandlermodul angeschlossen werden. Die Baugruppe ist als doppelseitig beschichtete Karte ausgeführt. *Tabelle 6* zeigt die Stückliste. Die ICs sollten in Fassungen gesetzt werden. *Bild 7* zeigt den Bestückungsplan. Die 16 Abblockkondensatoren (100 nF) sind auf dem Bestückungsplan mit B gekennzeichnet. Die Karte ist nicht sehr eng mit Bauteilen bestückt, so daß das Bestücken der Karte keine Schwierigkeiten machen

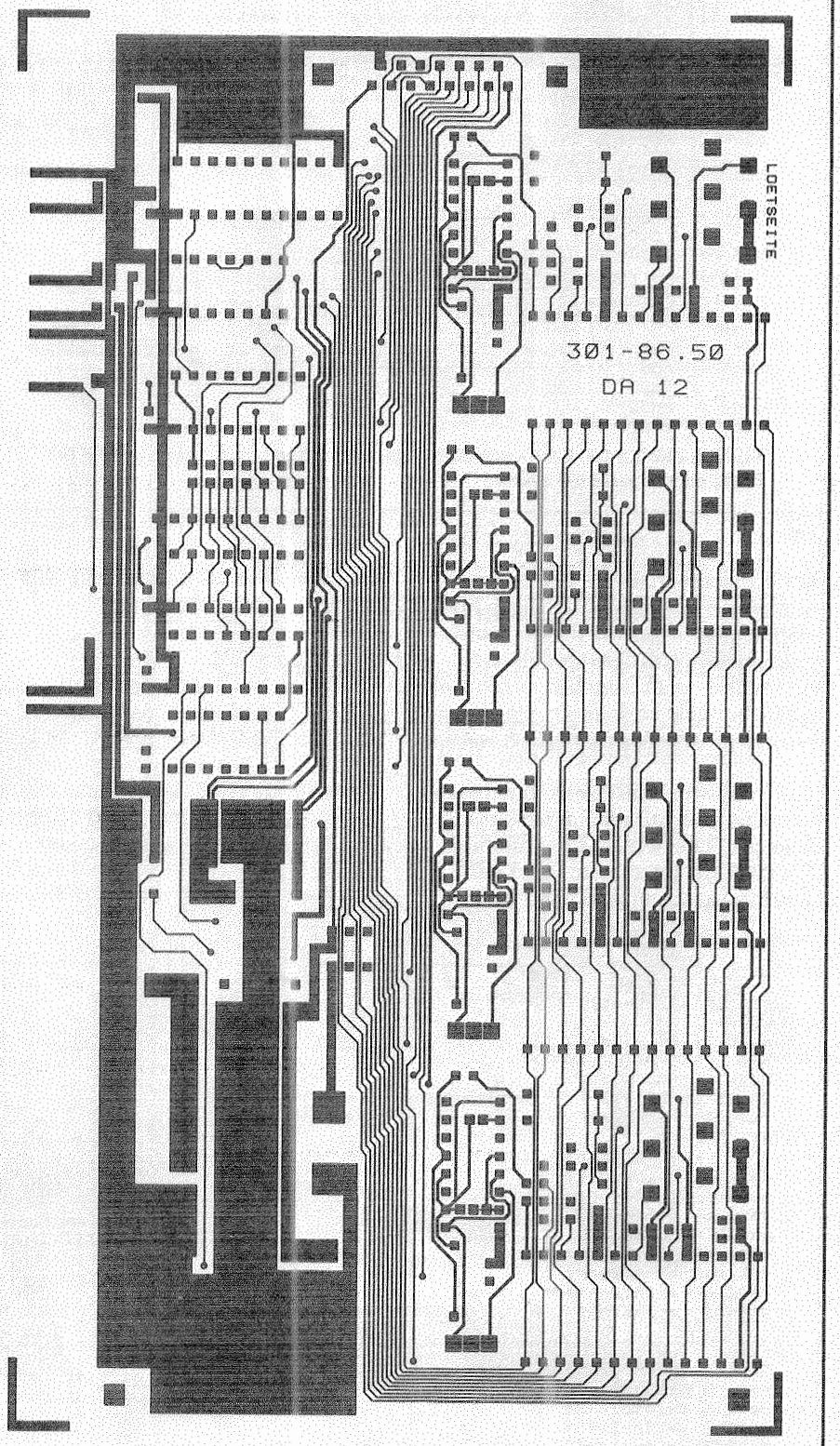


Bild 9. Die Lötseite

```
;*************************************************
;* Testprogramm DA12                             *
;* Erzeugt ein Saegezahnsignal am Ausgang von DAC1 *
;* 10.11.86 H.Assig                              *
;*************************************************

da12    equ     0310h   ; Basisadresse DA12
da1lsb  equ     da12
da1msb  equ     da12 + 1

stack   segment stack
        dw      20      dup(?)
stack   ends

code    segment para public 'code'

assume  cs:code
start:
        mov     ax,0
        mov     dx,da1lsb       ; Adresse DAC1
loop:
        out     dx,ax           ; Inhalt von AX an DAC1 ausgeben
        add     ax,16           ; und 16 aufaddieren
        jmp     loop
code    ends
end     start
```

Bild 10. Ein kurzes 8086-Assemblerprogramm erzeugt am Analogausgang der Karte eine Sägezahn-Funktion

dürfte. Insbesondere die Bestückungsseite (*Bild 8*) ist mit großflächigen Masseflächen versehen, um Störsignale abzuschirmen. Dank dem modularen Schaltungsaufbau brauchen nur so viele Kanäle bestückt zu werden, wie in der jeweiligen Anwendung benötigt werden.

Die Prozeßstromumsetzer können entfallen, wenn man nur die Spannungsausgänge benutzen will. In der Mimimalversion der Karte braucht nur ein Kanal ohne Stromausgang und ohne Spannungswandler bestückt zu sein. *Bild 9* zeigt die Lötseite der Karte.

Software

Die Ausgabe an den D/A-Wandler erfolgt als linksbündiges 16-Bit-Wort, da das Port-Bit D15 dem höchstwertigsten D/A-Umsetzer-Bit (DB11) entspricht. Das niederwertigste D/A-Umsetzer-Bit (DB0) ist Port-Bit D4. D0...D3 werden nicht berücksichtigt. Werte zwischen 0 und 4095 müssen daher mit 16 multipliziert werden. In dieser Form der Ausgabe kann man mit einem Byte-Zugriff die höchstwertigsten 8 Bit erreichen, um z. B. schnell Rampen zu durchfahren.

Das Beispielprogramm (*Bild 10*) in 8086-Assembler erzeugt ein Sägezahnsignal am Ausgang von DAC1, wenn die Baugruppe auf die Basisadresse 310H eingestellt ist. Die Signalamplitude entspricht dem eingestellten Ausgangsspannungsbereich an J2. Bei jedem Schleifendurchlauf wird 16 addiert, was einer 4-Bit-Linksverschiebung entspricht. Das Turbo-Pascal-Programm (*Bild 11*) liefert eine Folge von Sägezähnen an allen D/A-Ausgängen. Das Signal läßt sich am Oszilloskop gut betrachten.

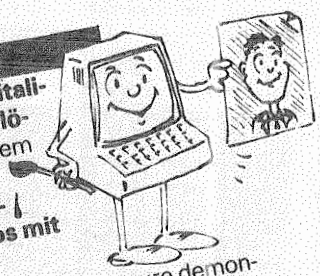
In Turbo-Pascal gestaltet sich die Ausgabe etwas schwieriger als in Assembler, da keine Wort-Zugriffe auf den E/A-Bereich möglich sind. Um einen 12-Bit-Wert ausgeben zu können, sind zwei Portanweisungen zu schreiben. Im Beispiel wird eine 8-Bit-Ausgabe verwendet, um das Programm übersichtlich zu halten. Das niederwertige Byte muß zuerst geschrieben werden, da mit dem höherwertigen Byte die Umsetzung gestartet wird. Auf einem 2-Kanal-Oszilloskop können schöne Bilder erzeugt werden, indem ein D/A-Kanal mit dem x-Eingang und ein anderer mit dem y-Eingang verbunden wird. Bei geschickter Programmierung lassen sich auch aus einer Hochsprache heraus bewegte Figuren darstellen.

```
program da12test;

(* 10.11.86 H.Assig *)

const
 da12  = $310;    (* Basisadresse der DA12 *)

var
 i       : integer;
 da1lo   : integer;
 da1hi   : integer;
 da2lo   : integer;
 da2hi   : integer;
 da3lo   : integer;
 da3hi   : integer;
 da4lo   : integer;
 da4hi   : integer;

begin
 da1lo := da12+0;
 da1hi := da12+1;
 da2lo := da12+2;
 da2hi := da12+3;
 da3lo := da12+4;
 da3hi := da12+5;
 da4lo := da12+6;
 da4hi := da12+7;
 clrscr;
 writeln('*********************************************');
 writeln('*                                           *');
 writeln('*                DA12-Test                  *');
 writeln('*                                           *');
 writeln('*********************************************');
 writeln;

 writeln;
 writeln('Liefert an den DAC-Ausgaengen ein Signal.');
 writeln;
 write  ('<Ende mit jeder Taste>');

   port[da1lo] := 0;
   port[da2lo] := 0;
   port[da3lo] := 0;
   port[da4lo] := 0;

 repeat
  for i := 0 to 255 do
   begin
    port[da1hi] := i;
    port[da2hi] := i;
    port[da3hi] := i;
    port[da4hi] := i;
   end;
 until keypressed;

    port[da1hi] := 0;
    port[da2hi] := 0;
    port[da3hi] := 0;
    port[da4hi] := 0;
end.
```

Bild 11. Sägezahn-Funktion in Turbo-Pascal programmiert

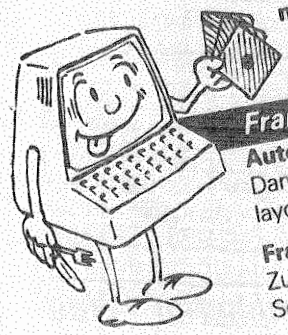
Hannover, CEBIT, Franzis.

Gespräche, Demonstrationen, Erfahrungsaustausch. Halle 6, Stand F20/22



Neues von mc:
Erleben Sie ein neues **Video-Digitalisierungssystem von hoher Auflösung.** An jedem Messetag auf dem Franzis-Stand.
Ihr ausgedrucktes Computer-Portrait nehmen Sie kostenlos mit nach Hause.

Außerdem: mc-Redakteure demonstrieren das **Betriebssystem OS-9 per mc-68000-System.**



Franzis-Software präsentiert:
Auto-Router 3
Damit erstellen Sie Ihre Platinenlayouts automatisch.

Franzis CAD
Zum preiswerten Selberzeichnen Ihrer Schaltpläne.

Im Dialog mit Fachbuchautoren

Buchautoren der Fachbereiche Elektronik und Computer und die Lektoren des Franzis-Verlages stehen Ihnen Rede und Antwort. **Am Samstag, den 7. März, den ganzen Messetag.**

Aktuelles aus der Funkschau:
Alles über LANs, WANs, ISDN, ISO und über moderne Telekommunikation im Funkschau-Sonderdruck. Kostenlos für Sie zum Mitnehmen.

Und nicht zuletzt:
Fachgespräche, Informationen, Fachbücher, Sonderpublikationen zu allen Bereichen der Elektronik und die Messeausgaben der Franzis-Zeitschriften erwarten Sie.

Gönnen Sie sich bei uns Ihre informative Messepause.
Wir freuen uns auf Sie.

Franzis'
Franzis-Verlag, Halle 6, Stand F 20/22.

mc-markt

C-64 — ohne Netz

Das Ingenieurbüro Drust hat einen interessanten Zusatz zum C-64 entwickelt, der den Computer vom 220-V-Netz unabhängig macht und einen Betrieb an 12 V Gleichspannung (z. B. Autoakku) erlaubt. Da die Erweiterung keine Einschränkung bezüglich der Softwareverträglichkeit bringen darf, wird die interne Uhr, die im Normalfall mit den 50-Hz-Impulsen des Netzes synchron läuft, nach dem Umbau durch einen quarzstabilen Takt ersetzt. Somit sind sogar die wenigen Programme, die auf diese Uhr zugreifen, lauffähig. Der Zusatz ist also kein simpler 12-V/220-V-Wandler, sondern eine echte DC-Versorgung, deren Inbetriebnahme auch eine Modifikation notwendig macht (Umverdrahtung, Einbau des Quarztaktes und Veränderung der Anschlußbuchse). Der Umbau wird für 68 DM vom Ingenieurbüro Drust durchgeführt, kann aber auch selbst problemlos vorgenommen werden. Die Kosten für den eigentlichen Gleichspannungs-Zusatz belaufen sich auf 50 DM. Nach dem Umbau ist der Betrieb des C-64 an der Steckdose via Netzteil nach wie vor möglich. Die gleiche Modifikation kann auch mit dem Floppy-Laufwerk 1541 vorgenommen werden.

Ingenieurbüro Drust
Darmstädter Str. 77
6103 Griesheim
☎ 0 61 55/22 63

Flinker Laser

In fünf Sekunden eine Schreibmaschinenseite – das Ganze in ausgezeichneter Darstellungsqualität und in einer Menge verschiedener Schriftarten – schafft der Toshiba-PageLaser 12. Er besitzt einen eingebauten Pufferspeicher von 512 KByte, emuliert zahlreiche Druckerstandards (Laser/Matrix/Typenrad) und kann auf Wunsch mit Papierführungen für die Ein- und Ausgabe von 250 bzw. 750 Blatt sowie mit einer Sortierautomatik ausgerüstet werden. Alles in allem ein attraktiver Drucker für typische Büro-Anwendungen.

Toshiba Europa GmbH
Hammer Landstr. 115
4040 Neuss 1
☎ 02101/1580

Gehört zu den schnellsten seiner Leistungsklasse: Toshibas PageLaser 12

Völkner

COMPUTER AKTUELL

Sensationelle Sonderangebote!

Apple IIc 798,-

Nutzen Sie jetzt Ihren sensationellen Preisvorteil durch unseren Großeinkauf!

Apple Profi-Computer „IIc":

128-kB-Arbeitsspeicher. Eingebautes 5¼"-Diskettenlaufwerk. 143-kB-formatiert. Englische Schreibmaschinen-Tastatur ohne deutsche Sonderzeichen. 63 Tasten. 94 Zeichen. 2 Sondertasten. 4 Cursortasten. Umschalter 40/80 Zeichen. Anschluß für 2. Disketten-Laufwerk. Gewicht nur 3,5 kg, daher ideal für Manager, Einkäufer, Reisende und alle, die ständig und überall neueste Informationen verarbeiten müssen. Anschluß 220 V/50 Hz. Inkl. engl. Bedienungsanleitung. Demo- u. Systemtools-Programm-Disketten. **Solange Vorrat!**

Best.-Nr. 9917352 ..... **nur 798,– DM**

Achtung, wichtiger Bestellhinweis!

Apple „IIc" ist **nur** im Verbund mit „Scribe-Printer" **oder** „Image-Writer" lieferbar.

Apple Thermo-Transfer-Drucker „Scribe-Printer":

80 Zeichen pro Sekunde! **80** Zeichen pro Zeile. Grafikauflösung max. 160×144 Punkte per Inch. Traktor, Endlos- und Einzelblatt. Lieferung inkl. englischem Anleitungs-Handbuch. Anschluß 220 V/50 Hz.

Solange Vorrat!

Best.-Nr. 9917361 ..... **nur 499,– DM**

Apple Profi-Drucker „Image-Writer":

120 Zeichen pro Sekunde! 136 Zeichen pro Zeile. Professionell mit 9×9 **Matrix**. Exzellentes Schriftbild. Lieferung inkl. engl. Anleitungs-Handbuch. Anschluß: 220 V/50 Hz.

Solange Vorrat!

Best.-Nr. 9917370 ..... **nur 749,– DM**

Achtung, wichtiger Bestellhinweis!

Apple „Scribe-Printer" und „Image-Writer" sind **nur** im Verbund mit Apple „IIc" lieferbar.

Apple Monochrom-9"-Monitor:

Professionelles Design. Hochauflösend **inkl. Ständer** und Anschlußkabel. Monitor grün. Anschluß 220 V/50 Hz. **Solange Vorrat!**

Best.-Nr. 9917389 ..... **nur 395,– DM**

Apple 2. Diskettenlaufwerk:

Zum Anschluß an „IIc". Speicherkapazität 143 kB formatiert. Vereinfacht die Übertragung von Informationen von einer Diskette zur anderen. Anschluß 220 V/50 Hz. **Solange Vorrat!**

Best.-Nr. 9917398 ..... **nur 448,– DM**

* ***Ihr Spezial-Programm für „IIc".***
* ***Umfaßt die häufigsten Anwendungen.***

Apple „Works":

Miteinander verknüpft sind leistungsfähige Textverarbeitung, umfangreiche Tabellenkalkulation und effektive Datenverwaltung. Für professionelles Arbeiten mit mehreren Unterlagen (Statistik, Briefe u.v.a.). Lieferung inkl. 3 Disketten (engl./US-Version).

Solange Vorrat!

Best.-Nr. 9917405 ..... **nur 98,– DM**

Apple ist eingetragenes Warenzeichen für APPLE-Computer-Inc.-Kalifornien!

Deutsche Fachliteratur für Apple „IIc".

Anwenderprogramme Apple „IIc":

Autor Floegel. Hofacker Verlag. Für Einsteiger und Fortgeschrittene. 130 Seiten mit zahlreichen Abbildungen.

Best.-Nr. 0403959 ..... **19,80 DM**

Apple „IIc" Benutzerhandbuch:

Autor Wassermann. Braucht jeder „IIc"-User. Inhalt: Bedienung, Soft-Basic-Erklärung, weitere Programmiersprachen, Anschluß externer Geräte u.v.a. 324 Seiten.

Best.-Nr. 0403977 ..... **35,– DM**

Völkner electronic GmbH & Co. KG

Versand und Zentrale: 3300 Braunschweig · Postfach 5320 · Telefon (0531) 87620 · Telex 952547

005

Wann steigen Sie ein?

Eine günstigere Gelegenheit gab es noch nie:

COMM… PLU…

Das komplett… mit Hard…

Denn die 4 wichti… sind bereits im Grun… Textverarbeitung, Da… und Geschäftsgrafik…

Unten:
HIGHSCREEN
Hochauflösender Datenmonitor DP 612 für alle Computer mit Video-ausgang

229.-

Die große Lösung:

Was kostet diese komplette Computeranlage mit Zentraleinheit, Diskettenlaufwerk, Monitor und Drucker?

Bei VOBIS nur

975.-

PLUS/4 ..... 198.-
Floppy 1551 ..... 299.-
HIGHSCREEN
DP 612 Monitor .. 229.-
MCS 801 Drucker . 399.-
Statt 1125.-
Komplett nur
975.-

Rechts:
COMMODORE MCS 801
8-farb-Matrixdrucker für PLUS/4 (und alle anderen COMMODORE VC-Computer). 80 Zeichen/Zeile für normal breites Papier

399.-

PLUS/4 ..... 198.-
Floppy 1551 299.-
Statt 497.-
Komplett nur
399.-

198.-

PLUS/4
64 K Speicher, umfangreiches BASIC, Farbe, Sound, Grafik. Incl. Text, Datei-Kalkulations- und Grafiksoftware: Anschließbar an normalen (Farb-) Fernseher oder Datenmonitor.

kompetent + preiswert

VOBIS

Deutschlands umsatzgrößter Microcomputer-Spezialist

WICHTIG! Herstellerbedingte Lieferzeiten! Aufgrund erhöhter Nachfrage ist nicht immer alles sofort lieferbar!

VERSAND-ZENTRALE:
Postfach 1778
Rotter Bruch 32-34
5100 AACHEN
☎ 0241/50 00 81
Tx 832389 vobis d

FILIALEN:
AACHEN Viktoriastr. 74 · 0241/54 31 00
AACHEN Pontstraße 60 · 0241/3 38 06
BERLIN 30 Kurfürstenstr. 101 · 030/2 13 94 80
BIELEFELD Herforder Str. 106 · 0521/6 38 78
BREMEN Violenstraße 37 · 0421/32 04 20
DORTMUND Hamburger Str. 110 · 0231/57 30 72
DÜSSELDORF Wielandstr. 21 · 0211/35 99 64
FRANKFURT Frankenallee 207/209 · 069/73 40 49
HAMBURG Krohnskamp 15 · 040/2 79 46 76
HANNOVER Berliner Allee 47 · 0511/81 65 71
KARLSRUHE Kriegsstr. 27/29 (am BGH) 0721/37 82 68
KONSTANZ Kreuzlinger Str. 18 · 07531/1 55 60
KÖLN Mathiasstr. 24-26 · 0221/24 86 42
MÜNCHEN Aberlestr. 3 · 089/77 21 10
NÜRNBERG Vordere Ledergasse 8 · 0911/23 29 95
STUTTGART Marienstr. 11-13 · 0711/60 63 36

Die preiswerte Lösung:

PLUS/4 mit Datenrecorder und Belegdrucker. Das alles schon unter 400.-!

Links: **COMMODORE 1520**
4-farb Drucker/Plotter. Für PLUS/4 (und alle anderen VC-Computer)

199.-

Rechts:
2-farb Drucker
CITIZEN IDP 560
40 Zeichen. Für PLUS/4 (und alle anderen VC-Computer)

99.-

59.-

oben: **1531 Datenrecorder** für PLUS/4 und alle anderen VC-Computer

PLUS/4 ..... 198.-
1531 Recorder 59.-
Statt 257.-
Incl. BASIC-Lernkurs auf Programm-Kassette
Komplett nur
229.-

DER VOBIS-PLUSPUNKT
Einsenden an VOBIS, Postfach, 5100 Aachen
Hiermit bestelle ich:

Meine Adresse:

© pSE-DESIGN